State of Michigan (SOM)

Systems Engineering Methodology Version 1.1

The Systems Engineering Methodology (SEM) of the State Unified Information Technology Environment (SUITE)

September 2007

Michigan Department of Information Technology

PREFACE

This initial development of the *State of Michigan Systems Engineering Methodology (SEM)* was published in April 2007, and was performed as part of a continuing effort to improve the quality, performance, and productivity of State of Michigan information systems. Development of the SEM was governed by the Michigan *State Unified Information Technology Environment* (SUITE) initiative.

The purpose of SUITE is to standardize methodologies, procedures, training, and tools for project management and systems development lifecycle management throughout the Michigan Department of Information Technology (MDIT) in order to implement repeatable processes and conduct development activities according to Capability Maturity Model Integrated (CMMI) Level 3 requirements. A formal enterprise level support structure will be created to support, improve and administer SUITE, SEM, Project Management Methodology, and related enterprise initiatives. Until that structure is in place, questions regarding SEM should be sent to the SUITE Core Team at SUITE@michigan.gov where they will be addressed by a rotating matrixed team.

This SEM replaces in total the former State of Michigan Systems Development Lifecycle (SDLC) document dated November 2001 and related templates.

ACKNOWLEDGEMENTS

The State of Michigan would like to thank the following individuals and organizations that made this version of the State of Michigan Systems Engineering Methodology possible. Without their input, dedication and hard work, this would not have been achieved.

MICHIGAN DEPARTMENT OF INFORMATION TECHNOLOGY (MDIT) PROJECT SPONSORS AND MANAGEMENT	
Teresa M. Takai, Director-MDIT and State of Michigan Chief Information Officer	Kenneth D. Theis, Senior Chief Deputy Director-MDIT
Carol Steffanni, Deputy Director-MDIT, Bureau of Strategic Policy	Lynn Draschil, Senior Deputy Director-MDIT Bureau of Agency Services
Joel Storchan, Director-Office of Standards and Contract Management, MDIT Bureau of Strategic Policy	C. Douglas Couto, Information Officer-MDIT, Agency Services-Transportation, Management and Budget, Civil Service, State Budget Office
Dave Borzenski, State Department Administrator, MDIT Agency Services- Department of Treasury	Scot Ellsworth, State Department Administrator, MDIT Office of Enterprise Architecture
Dan Buonodono, Project Management Specialist, MDIT Project Management Resource Center	Virginia Hambric, State Division Administrator, MDIT Agency Services-Human Services and MiCSES

PRODUCTION RELEASE (April 2007)	
Dan Buonodono, Project Management Specialist, MDIT Project Management Resource Center	Virginia Hambric, State Division Administrator, MDIT Agency Services-Human Services and MiCSES
Leigh A. Scherzer, Account and Project Manager, Dedicated Customer Unit, MDIT Agency Services-Department of Labor & Economic Growth. SEM	W. Steve Wensko, Operations Manager, Operations Systems Section, MDIT Agency Services-Transportation
Paul Perla, Team Lead, MDIT-Employee and Financial Services, Human Capital Management	James “Rock” Rakowski, State Administrative Manager, Agency Liaison Section, MDIT Office of Enterprise Security
Shawn M. Bauman, Special Projects Analyst, MDIT Agency Services-Department of Human Services and MiCSES	Donna Sivaraman, Programming Services Manager, MDIT Agency Services-Treasury
Kyle Wilson, IT Specialist, Quality Assurance Team, MDIT Agency Services-Transportation	Jerry Morey, State Administrative Manager, Planning and Solutions Development, MDIT Infrastructure Services – Data Center Operations
Samuel Roberts, IT Manager, Application Development, MDIT Agency Services-Natural Resources and History, Arts & Libraries	Mark Breithart, IT Specialist, Application Development, MDIT Agency Services-Environmental Quality
Fred Moye, IT Programmer/Analyst, MDIT Agency Services-Michigan State Police	Sandy Cain, Forms Officer, MDIT Office Automation-Technical Training
Robert Surber, State Division Administrator, MDIT-Center for Geographic Information	Dave Archer, Manager, Design Section, Infrastructure Services, MDIT Office Automation-Design and Delivery Division

Dave Reicosky, Project Manager, MDIT Infrastructure Services – Telecommunications	P. Michael Spagnuolo, IT Manager, Application Programming Support Services, MDIT Agency Services-Department of State
Randy Leyrer, Development Services Manager, MDIT Agency Services-Department of Treasury	Lucy Pline, Technical Manager, MDIT Agency Services-e-Michigan Web Development Division
Allan DeKoninck, PMP, IT Manager, Data Warehouse and Database Team, MDIT Agency Services-Community Health	Patty Whitlock, Departmental Analyst, Analysis & Review Team, MDIT Agency Services-Human Services and MiCSES
Tanis S. Lerash, PMP, IT Manager, MDIT Agency Services-Civil Service, DMB, DMB Retirement	Sue Tomes, Departmental Manager, Analysis & Review Team, MDIT Agency Services- Human Services and MiCSES
Rose Johnson-King, Information Security Specialist, MDIT Office of Enterprise Security-Communications, Awareness, Homeland Security	Michael Shanahan, Director, MDIT Agency Services-e-Michigan Web Development Division
Bryan Farr, IT Specialist, Application Development and Support, MDIT Agency Services-Michigan State Police and Department of Military and Veterans Affairs	Steve Ezzo, IT Specialist, MAIN-FACS Section, MDIT Agency Services-Management and Budget
Kirt Berwald, Director, Field Services Division, MDIT Infrastructure Services	Terry O'Neill, IT Manager, Business Services, MDIT Infrastructure Services
Aparna Agrawal, Director, Technical Services, MDIT Infrastructure Services	Sara Kanya, State Administrative Manager, Enterprise Platform Services, MDIT Infrastructure Services – Data Center Operations
Bob McDonough, State Administrative Manager, MDIT Office of Enterprise Architecture	Benjamin T. Kinsey, Information Systems Analyst, Project Development Section, MDIT Agency Services-Transportation

ORGANIZATIONS
STATE OF MICHIGAN – DEPARTMENT OF INFORMATION TECHNOLOGY
U.S. DEPARTMENT OF ENERGY – OFFICE OF THE CHIEF INFORMATION OFFICER

The SEM Development Teams owe a large debt to Brenda Coblentz of the U.S. Department of Energy (DOE) for both her encouragement in our efforts and for permitting us the free use of the DOE's own CMMI Level 3 compliant SEM as a basis for this document. In particular, much of this document draws directly from the DOE's Systems Engineering Methodology, which as of March 30, 2007 can be found at (http://cio.energy.gov/documents/SEM3_1231.pdf).

Chapter Page

Chapter Page

Exhibits **Page**

Industry Sites

Carnegie Mellon University Software Engineering Institute..... http://www.sei.cmu.edu

International Council on Systems Engineering.....http://www.incose.org

Project Management Institute .....http://www.pmi.org

Quality Assurance Institute.....http://www.qaiworldwide.org/

[1] Addresses are as of February 2007

Chapter: **1.0 Introduction**

Description: The *Systems Engineering Methodology (SEM) of the State Unified Information Technology Environment (SUITE)* provides guidance for information systems engineering related project management activities and quality assurance practices and procedures. The primary purpose of the methodology is to promote the development of reliable, cost-effective, computer-based solutions while making efficient use of resources. Use of the methodology will also aid in the status tracking, management control, and documentation efforts of a project.

Development of the SEM was governed by the Michigan *State Unified Information Technology Environment* (SUITE) initiative.

The purpose of SUITE is to standardize methodologies, procedures, training, and tools for project and systems development lifecycle management throughout the State of Michigan Department of Information Technology (MDIT) in order to implement repeatable processes and conduct development activities according to Capability Maturity Model Integrated (CMMI) Level 3 requirements.

This information system engineering methodology is consistent with other methodologies used in State and Federal Governments and private industry. It complies with State of Michigan policy on project management, software configuration management, security, and records management. It should be used in conjunction with all State of Michigan information management programs and initiatives.

It is important to differentiate between a project management methodology and a system engineering methodology. A project management methodology covers all the things a project manager needs to do regardless of whether the project is a software development, package selection, or relocation of a work unit.

The State of Michigan Project Management Methodology (PMM) covers standard areas of project management (Cost Management, Risk Management, Scope Management, Resource Management, Communications Management, Quality Management, Time Management, Procurement Management, and Integration Management) and purposely does not include the separate concepts and requirements of system engineering, leaving that to be included in the SEM. Conversely, this SEM does not reiterate the standards of project management, instead referring to the PMM as appropriate.

The PMM is the methodology for management of the work effort. The SEM is the step-by-step development of the software application.

The State of Michigan has a consistent project management methodology in place which can be used for all types of projects. The State of Michigan now also has a consistent system engineering methodology that is a companion to the project management methodology.

In this way, people can move comfortably from applications development, to infrastructure roll out, to software selection to even relocating to new buildings using the same approach throughout the organization.

Significant input for the methodology was obtained from information management programs throughout the State. The methodology integrates State of Michigan best practices and focuses on the quality of both the systems engineering process and the work products generated from the process.

The SEM is derived from the principles and standards advocated by information management industry leaders, such as The Institute of Electrical and Electronics Engineers (IEEE), the Carnegie Melon Software Engineering Institute (SEI), and the Department of Energy (DOE). ***This methodology is designed to enable State of Michigan project teams to achieve Level 3 maturity on the SEI Capability Maturity Model Integrated (CMMI).***

Quality assurance is integrated into the methodology, making quality the responsibility of all project managers and team members. To assure the development of quality products, the methodology prescribes reviews, inspections, and audits for the lifecycle processes and technical work products. To protect the integrity of information systems, the methodology also prescribes configuration controls over system components, data, and technical documentation.

The methodology encompasses the aspects of the information systems engineering project lifecycle, from project planning through production and maintenance, and integrates basic lifecycle management concepts (*Exhibit 2.0-1 SEM Overview Diagram* on page 13).

The SEM is intended to be used by individuals, project teams, and managers who are responsible for developing a new computer-based solution or effecting changes to an existing system. The methodology, including its templates, is reviewed on a regular basis and will be modified as needed to keep pace with the changing needs of State of Michigan information systems engineering environment and the continuing technical advances in the information systems industry. As a result of the reviews, it is anticipated that a new release of the SEM will be issued within 12 months of the initial release date.

The following sections provide additional information about using the SEM.

Section: **1.1 Enterprise Implementation of the Methodology**

Description: While the focus of the SEM is at the project level (see Section 1.2, Project Implementation of Methodology), it is recognized that there must be an enterprise-wide ability for managing information systems development, integration, and maintenance processes and quality oversight to ensure the delivery of high-quality products. Within this context, an agency is a State of Michigan unit, (e.g., a State of Michigan department or bureau, within which, generally, many projects are managed). The SEM integrates systems and infrastructure project management and quality assurance practices and is designed to be flexible. It can be adapted to accommodate the specific needs of any information systems engineering organization and all computing platforms used in the State. With adoption of the SEM as the State of Michigan standard process for developing and maintaining systems, any additional specific or unique management processes should be integrated into the organization to help project managers and technical staff perform more effectively.

In a mature organization, the processes are institutionalized. They are documented, reusable, and consistent with the way the work is actually accomplished. The process definitions are updated when necessary, and improvements are applied when appropriate, with broad-based active involvement across the organization. Roles and responsibilities are clear and communicated throughout projects and across the organization. Enterprise-wide training ensures personnel are well trained so they can perform their roles effectively and efficiently.

The following tasks describe processes and activities complementary to those at the project level and aimed at maturing the entire enterprise in terms of capability to deliver high quality products.

Resource: Carnegie Mellon University, Software Engineering Institute, *Capability Maturity Model: Guidelines for Improving the Software Process*, Addison Wesley Longman, Inc., 1994

Tasks: The following tasks are involved in enterprise implementation of the SEM.

1.1.1 Enterprise Process Management
1.1.2 Enterprise Curriculum
1.1.3 Quality Oversight

Task: **1.1.1 Enterprise Process Management**

Description: The goal of this task is to establish the State's responsibility for lifecycle process activities that improve its overall capability. The State provides long-term commitments and resources to coordinate the development and maintenance of the process across current and future projects.

With MDIT's adoption of the SEM, it became the state-wide enterprise process for systems development. The SUITE Core Team will periodically assess its processes and develop an action plan for improvement. Changes to the process are then communicated to those individuals within the State responsible for implementing the process.

New processes, methods, and tools in limited use in the State are to be monitored, evaluated, and, where appropriate, transferred to other parts of the MDIT organization. The major component of Enterprise Process Management is the Enterprise Repository.

Enterprise Repository

It is anticipated that the SUITE Core Team will create, manage and control a repository to collect and make available data on the systems engineering process and resulting work products, (e.g., productivity data, quality measurements, and estimates of size, effort, and cost). It is anticipated that the repository will serve to improve project management planning and estimating by providing a resource for future systems engineering efforts. The repository will also establish and control a statewide library of systems process-related documentation including policies and procedures that will serve as a path to achieving CMMI Level 3. The library will be cataloged for easy reference and the contents made available for use by project teams and other systems-related groups. Library contents are to be updated as appropriate.

Work Products: An action plan is to be developed based on the periodic assessments. The action plan will identify guidelines for implementing the changes to address specified assessment findings and assigns responsibility for implementing changes.

An improvement plan is to be developed and maintained for process development and improvement activities. The plan uses the action plan and other improvement initiatives as primary input. The plan defines and schedules activities to be performed, assigns responsibility and identifies resources required for implementing the plan.

It is anticipated that a repository will be established for enterprise process and data (metrics) information. DIT staff will be trained in the use of, and have controlled access to, the repository.

Review Process: Conduct structured walkthroughs for each of the written work products to remove as many defects as possible.

Task: **1.1.2 Enterprise Curriculum**

Description: An organization that is well prepared for the challenges posed by information systems engineering projects must ensure that its personnel are well trained to perform their roles effectively and efficiently. The goal of this task is to describe the areas of training that must be addressed to ensure the State of Michigan has a documented process in place to manage training activities on an ongoing basis.

The process shall be based on documented enterprise training standards. The standards should include how courses are to be developed (or standards that must be met where courses are procured) and how they are to be maintained according to these standards. Members of the training group (or vendors if training is acquired) need to have the necessary skills and knowledge to perform their training activities.

When determining the skills and knowledge needed for a project, the project teams are responsible for identifying their unique needs. Each project needs to evaluate its current and future skill needs and determine how these skills will be obtained. Some skills may be imparted through informal vehicles (e.g., on-the-job training, mentoring,) while other skills may need more formal training vehicles (e.g., classroom, self-study.) Appropriate vehicles need to be selected and used.

Responsibility for training needs to be identified and communicated. It may lie with a single manager within the organization, or may be shared by several managers, each responsible for one or more knowledge areas or subjects. The specific enterprise responsibility for training needs to be identified, documented, and available for viewing by staff.

Waiver Process: A waiver procedure for required training needs to be established and used to determine whether staff already possesses the knowledge and skills to perform their jobs.

Measurements: Measurements need to be identified, collected, and used to assess the status of training activities. Measurements should address areas such as the quality of the training, and if it meets the needs of the staff. Measurements and the enterprise training should be reviewed with management on a regular basis. (In this context, “measurement” is not meant to be CMMI Level 4.)

Work Products: A written training policy describing how the State of Michigan will meet training requirements needs to be developed, communicated, and followed. The policy needs to be periodically reviewed and revised as appropriate based on feedback collected.

A written training plan that addresses how the training needs of the State of Michigan will be met. The plan should include information such as how training needs will be identified, what training is required, how training will be delivered, the cost and resources required, enterprise placement of the training function,

who will be involved, when and how the plan will be reviewed and revised, and a work breakdown structure that identifies all of the activities involved.

Maintain records that training has been conducted and completed waivers, if and where appropriate.

Note: A written training plan should be developed for each project and a training program should be developed for system implementation and operation.

Review Process: Conduct structured walkthroughs for each of the written work products to remove as many defects as possible.

Task: **1.1.3 Quality Oversight**

Description: The goal of this task is to establish the enterprise responsibility for the quality oversight of information technology investments. While the lifecycle process activities for project implementation and maintenance are documented within the lifecycle stages of the SEM, an enterprise quality oversight program provides long-term commitments and resources to coordinate the quality activities across current and future projects.

The quality oversight program should implement the appropriate level of management effort, and assume responsibility, accountability, and oversight for continued quality management process compliance within the organization. The quality oversight program should identify standards and best practices for product development, and ensure appropriate safety and security controls are in place, are effective, and reflect current accepted industry practices. The program should also ensure that project teams are aware of current State of Michigan computer and cyber security directives and have coordinated the project with computer security staff.

Work Products: A written quality oversight program describing how the organization will ensure the development of high quality information technology investments needs to be developed, communicated, and followed. The program needs to be periodically reviewed and revised as appropriate based on feedback collected.

The program should identify a point of contact for managing quality oversight and ensuring project risk assessments are conducted to determine the appropriate level of quality assurance activities to be applied. The program should ensure the level of quality assurance is tailored to the site and project needs. The oversight program should oversee the development and implementation of quality assurance processes and procedures, and ensure the development and implementation of project quality assurance plans and production and delivery of quality products.

Review Process: Conduct a structured walkthrough of the quality oversight program to remove as many defects as possible.

Section: **1.2 Project Implementation of Methodology**

Description: SUITE will integrate information systems engineering, project management and quality assurance practices and is designed to be flexible. It can be adapted to accommodate the specific needs of any information systems project and all computing platforms used in the State of Michigan including standalone and networked mainframes, servers, desktops, and other computers.

Projects that were initiated prior to the awareness or usage of this document should plan to implement the methodology at the earliest feasible stage or the next release of the product. If a Project Plan already exists, make the revisions necessary to integrate the systems engineering, project management, and quality assurance practices, as appropriate. If a Project Plan does not exist, develop a plan that summarizes the activities and deliverables of the previous stages and incorporates the methodology activities and products into the subsequent stages.

The information systems engineering methodology presented here does not supersede, replace, or override more stringent requirements that may apply to specific projects such as scientific and technical practices, and security and safety issues.

Questions: If specific questions are generated concerning the interpretation or applicability of portions of the methodology, the project team should attempt to resolve them during the project review activities built into the stages of the lifecycle. The system owner/user(s) and other project stakeholders must concur with any adaptations that are made.

When questions about interpretation or applicability of the guidance to a specific project cannot be resolved by the project team, the issue should be submitted to the site authority for information systems engineering, such as the team leader, supervisor, manager, or MDIT Client Services Director, for advice or resolution. SUITE Core Team staff may also be consulted on the interpretation or applicability of the methodology by sending e-mail to SUITE@michigan.gov.

Section: **1.3 Submitting Change Requests**

Description: The SEM environment is continuously changing as emerging technologies are integrated into projects, system owner/user requirements are expanded, and enterprise needs evolve. The SEM will be revised, as needed, to reflect changes in the environment, improvements suggested through user feedback, and the maturation of information systems engineering capabilities.

Users of the methodology are encouraged to submit suggestions for improving its content and to report any practices that are difficult to understand or create an implementation problem for a project team.

Suggestions and problems should be submitted on the SEM Change Request Form DIT-0181 that is available on the DIT TechTalk and SUITE websites. The form contains the accompanying instructions for guidance on completing this form.

The SEM Change Request Form should be submitted to the SUITE Core Team via e-mail at SUITE@michigan.gov. All requests will be evaluated and the originator of the request will be notified of the action taken.

Some requests will be handled immediately while others may require investigation by an ad hoc working group of knowledgeable personnel. In some cases, a request may not be appropriate for the current environment, but will be retained for future consideration.

Chapter: **2.0 Lifecycle Model**

Description: This chapter describes the lifecycle model used for the SEM. This model partitions the information systems engineering lifecycle into seven major stages, as shown in *Exhibit 2.0-1, SEM Overview Diagram* on page 13. Each stage is divided into activities and tasks, and has a measurable end point (Stage Exit). The execution of all seven stages is based on the premise that the quality and success of the product depends on a feasible concept, comprehensive and participatory project planning, commitments to resources and schedules, complete and accurate requirements, a sound design, consistent and maintainable construction techniques, and a comprehensive testing program. The lifecycle stages and activities are described in the following chapters.

Intermediate work products are produced during the performance of the activities and tasks in each stage. These work products are inspected and can be used to assess system integrity, quality, and project status. As a result, adequacy of requirements, correctness of designs, and quality of the products become known early in the effort.

At least one time for each work product, a Structured Walkthrough (SWT) is performed. A Structured Walkthrough is an organized procedure for reviewing and discussing the technical aspects of systems or software engineering work products including documentation. The walkthrough is usually conducted by a group of peers and may include reviewers outside the developer's immediate peer group. The *Structured Walkthrough Process Guide* provides detailed process information. This document is available on the MDIT SUITE website.

At the conclusion of each stage, a Stage Exit is initiated to review the work products of that stage and to determine whether to proceed to the next stage, continue work in the current stage, or abandon the project. The approval of the system owner and other project stakeholders at the conclusion of each stage enables both the system owner and the project manager to remain in control of the project throughout its life, and prevents the project from proceeding beyond authorized milestones. The *Stage Exit Process Guide* provides detailed process information. This document is available on the MDIT SUITE website.

The end products of the lifecycle are the information system product, the data managed by the system, associated technical documentation, and user training and support. The end products and services are maintained throughout the remainder of the lifecycle in accordance with documented configuration management procedures.

The lifecycle model provides a method for performing the individual activities and tasks within an overall project framework. The stages and activities are designed to follow each other in an integrated fashion, whether the stages of development are accomplished sequentially, concurrently, or cyclically. Project teams have the flexibility to adapt the lifecycle model to accommodate a

particular development methodology (e.g., spiral development,) information systems engineering technique (e.g., prototyping and rapid application development,) or other project constraints.

The amount of project and system documentation required throughout the lifecycle depends on the size and scope of the project. System documentation needs to be at a level that allows for full system operability, usability, and maintainability. Typically, projects that require at least one work-year of effort should have a full complement of documentation. For projects that require less than one work-year of effort, the project manager and system owner should determine the documentation requirements. In addition, the project's security and quality assurance criteria may require the performance of other activities and the generation of additional documentation.

The requirements for documentation should not be interpreted as mandating formal, standalone, printed documents in all cases. Progressive documents that continuously revise and expand existing documentation, online documents, forms, reports, electronic mail messages, and handwritten notes (e.g., informal conference records) are some examples of alternative documentation formats. Project managers should verify documentation standards within their sites.

The following sections provide additional information about the lifecycle model.

Bibliography: The following materials were referenced in the preparation of this chapter.

1. SLC Roadmap Document, Electronic Data Systems, Inc., 1991

2. Booch, G., *Object-Oriented Analysis and Design,* 2nd edition, Benjamin Cummings, 1994.

3. Budd, T., *An Introduction to Object-Oriented Programming*, 2nd edition, Addison-Wesley, 1996.

4. Carnegie Mellon University, Software Engineering Institute, *Capability Maturity Model: Guidelines for Improving the Software Process,* Addison Wesley Longman, Inc., 1994.

5. Carney, D, & Oberndorf, P. "The Commandments of COTS: Still Searching

for the Promised Land." Crosstalk 10, 5 (May 1997): 25-30.

6. Federal Acquisition Regulations. Washington, DC: General Services Administration, 1996.

7. Jacobson, I., *Object-Oriented Software Engineering,* Addison-Wesley, 1992.

8. Meyers, Craig & Oberndorf, Tricia. Open Systems: The Promises and the Pitfalls. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1997.

9. Open Systems Joint Task Force Baseline Study, 1996 [online].

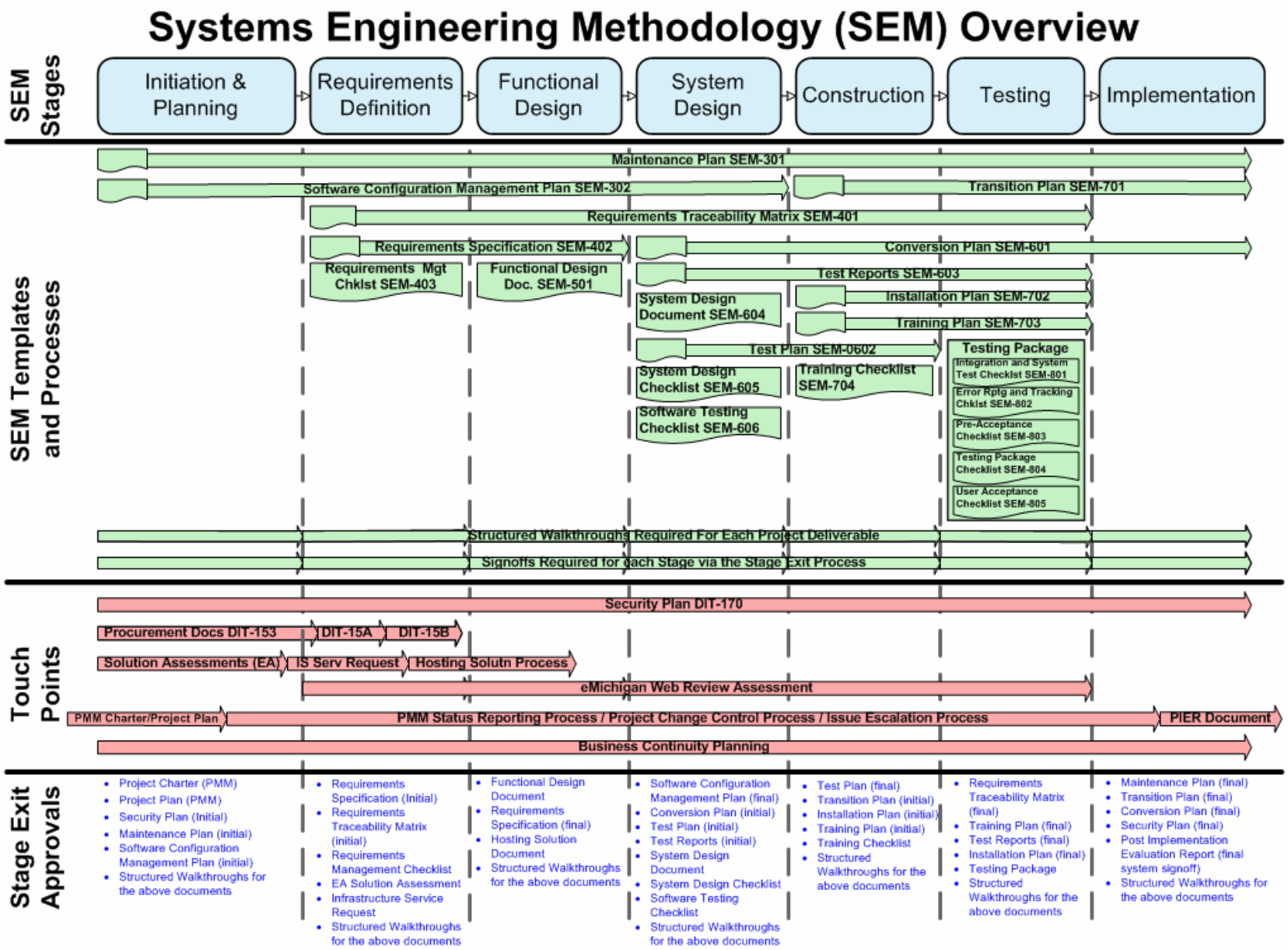
10. Open Systems Joint Task Force Case Study of U.S. Army Intelligence and Electronic Warfare Common Sensor (IEWCS), 1996 [online].

11. Pressman, Roger S., *Software Engineering - A Practitioner's Approach,* 4th edition, McGraw-Hill Companies, Inc., 1997.

12. Siy, Harvey, *Identifying the Mechanisms to Improve Code Inspections Costs and Benefits,* 1996 [online].

13. Yourdon, Edward, *Structured Walkthroughs,* second edition, Yourdon Inc., New York, 1978.

Exhibit 2.0-1 SEM Overview Diagram



Section: **2.1 Project Sizes**

Description: The lifecycle model used in this information systems engineering methodology can be applied to projects of varying sizes. In this model, projects are divided into three sizes: large, medium, and small. Each project size uses the same lifecycle stages. Medium and small projects may compress or combine stages and required documentation in direct proportion to the size of the development effort. The major differences between project sizes are determined by the following items.

- The estimated total labor hours (the level of effort) required to complete the project.
- The use of cutting edge or existing technology.
- The type and extent of both user and system interface requirements.
- The project's contribution to, and impact on, the activities carried out by the system users and other Departmental organizations.

The requirements, constraints, and risks associated with the project also influence the determination of project size. The project size and any plans for adapting the lifecycle model are documented in the Project Plan, which is reviewed and approved by the system owner and other project stakeholders.

The following subsections provide descriptions of the three project sizes used in this lifecycle model. *Exhibit 2.1-1, Information Systems Project Sizes*, shows the level of effort and complexity measures used to define the three sizes.

Large Projects: Large information systems engineering projects are included in the system owner's organizational long-range plans. Department-wide and site-specific projects are usually developed as large-sized projects and are likely to require a major acquisition of hardware and software. Typically, the larger the size and scope of the project, the greater the detail and coordination needed to manage the project. As risk factors and levels of effort increase, the scope of project management also increases and becomes a critical factor in the success of the project.

Medium Projects: Medium information systems engineering projects require less effort than large projects, typically use existing hardware and software, and might not be captured during the organizational long-range planning process. They are frequently developed to automate operations within a programmatic office or among a limited number of sites, and may be used to interface with other systems. Planning medium size projects within the context of the system owner organization's overall mission, and building in compatibility to the Departmental IT environment can improve the product's ability to interface with other users, organizations, and applications; and increase the product's longevity.

Small Projects: Small information systems engineering projects require minimal effort and use existing hardware and software. The operational details of a small project can easily be managed by the project manager, so formal documentation requirements are limited. A project is small when the system being developed will have limited functionality and use, meets a one-time requirement, or is developed using reusable code.

Exhibit 2.1-1 Information Systems Project Sizes

Complexity (and associated characteristics)	**Effort Required** (in staff months)		
	0-8	**9-24**	**25-n**
Low: - Existing or known technology - Simple interfaces - Requirements well known - Skills are available	Small	Small	Medium
Medium: - Some new technology - Multiple interfaces - Requirements not well known - Skills not readily available	Small	Medium	Large
High: - New technology - Numerous complex interfaces - Numerous resources required - Skills must be acquired	Medium	Large	Large

Note: Size is used as a guide to help determine the appropriate degree of project management, and whether any stages may be combined for a given effort. Within this context, size is a combination of level of effort required (all activities) and complexity of the requirements. Attributes of complexity include technology, team skills, interfaces, and level of understanding of requirements. Other factors that can influence adaptation include risk, visibility, and business impact.

Section: **2.2 Adapting the Lifecycle**

Description: The SEM implements well-defined processes in a lifecycle model that can be adapted to meet the specific requirements or constraints of any project. This section provides guidelines for adapting the lifecycle processes to fit the characteristics of the project. These guidelines help ensure that there is a common basis across all projects for planning, implementing, tracking, and assuring the quality of the work products.

The lifecycle model has built-in flexibility. All of the stages and activities can be adapted to any size and scope information systems engineering project. The lifecycle can be successfully applied to development projects, maintenance or enhancements, and customization of commercial software. The lifecycle is appropriate for all types of administrative, business, manufacturing, laboratory, scientific, and technical applications. For scientific and technical projects, adaptations to the lifecycle may be dictated by the project stakeholders or the requirements for reporting technical results in formal reports or journal articles.

Adaptations: The lifecycle can be compressed to satisfy the needs of a small project, expanded to include additional activities or work products for a large or complex project, or supplemented to accommodate additional requirements, (e.g., security requirements). Any modifications to the lifecycle should be consistent with the established activities, documentation, and quality standards included in the methodology. Project teams are encouraged to adapt the lifecycle as long as the fundamental information systems engineering objectives are retained and quality is not compromised.

The following are some examples of lifecycle adaptations:

- Schedule stages and activities in concurrent or sequential order.
- Repeat, merge, or simplify stages, activities, or work products.
- Include additional activities, tasks, or work products in a stage.
- Change the sequence or implementation of lifecycle activities.
- Change the development schedule of the work products.
- Combine or expand activities and the timing of their execution.

The lifecycle forms the foundation for project planning, scheduling, risk management, and estimation. When a lifecycle stage, activity, or work product is adapted, the change must be identified, described, and justified in the Project Plan. The Project Plan is developed as a separate document and includes a description of the systems development lifecycle, which is the organization's standard process.

Exhibit 2.2-1, Adapting the Lifecycle, shows how stages can be combined to accommodate different size projects and information systems engineering techniques. *Notes* are provided throughout the lifecycle stage chapters to identify activities that have built-in project adaptation strategies. Adaptations should not introduce an unacceptable level of risk and require the approval of the system owner and other project stakeholders.

When adapting the lifecycle model, care must be taken to avoid the following pitfalls:

- Incomplete and inadequate project planning.
- Incomplete and inadequate definition of project objectives and requirements.
- Lack of a development methodology that is supported by information systems engineering preferred practices and tools.
- Insufficient time allocated to complete design before coding is started.
- Not defining and meeting criteria for completing one lifecycle stage before beginning the next.
- Compressing or eliminating testing activities to maintain an unrealistic schedule.

Sample Statements:

The following are sample statements that can be used in the Project Plan to describe different types of lifecycle adaptations. The first example shows a scenario where the Concept Document will not be developed in the Initiation and Planning Stage.

A Concept Document will not be developed for this project. The need for the product has been documented in several organizational reports and was included in the fiscal year long-range plans. The platform for the project is currently used for all applications owned by this organization. There are no known vendor packages that will satisfy the functional requirements described by the system owner.

The following is a sample statement that shows how work products from two different stages can be combined into one deliverable.

The Functional Design and System Design documents will be combined into one design document. A Stage Exit will be conducted when the design document is completed. To reduce the risk associated with combining the two documents, the project team will develop prototype screens and reports for review and approval by the system owner/user(s) as the prototypes are developed.

The following is a sample that shows how the seven lifecycle stages can be compressed into five stages for a small project.

This project will require 10 staff months of effort to enhance an existing application. The seven stages in the lifecycle will be combined into five stages as follows: (1) Initiation and Planning, (2) Requirements and Design, (3) Construction, (4) Testing, and (5) Implementation.

The following deviations will occur for document deliverables:

- *A Concept Document and a Business Case will not be necessary due to the restricted software and hardware platform.*
- *The Requirements Specification will be limited to the statement of enhancement requirements.*
- *The Functional Design and System Design documents will be combined into one design document.*
- *An amendment package will be developed for the existing Users Manual.*

Exhibit 2.2-1 Adapting the Lifecycle

Large Project

Initiation & Planning	Requirements Definition	Functional Design	System Design	Construction	Testing	Implementation
			Iterative Development[1]			

Medium Project

Initiation & Planning	Requirements Definition / Functional Design	System Design / Construction	Testing	Implementation
Rapid Prototyping[2]				

Small Project

Initiation & Planning / Requirements Definition / Functional and System Designs	Construction and Testing	Implementation

Project Management

Less More

Structured Walkthroughs should be performed for each major deliverable. Stage Exits should occur at the end of each stage.

Note: Iterative development and rapid prototyping are optional techniques that can be used on any size project.

[1] Each iteration produces working function(s) from integrated program modules.
[2] Iterations may produce any or all of requirements, system architecture, functional design, system design.

***Section*: 2.2.1 Tailoring Guidance**

Due to the large variation among system size and complexity, there is a need to offer guidance to the project / development manager regarding which components of the methodology, both project-based and product-based, are required.

The intent of this section is to provide flexibility in utilizing SEM components in the systems development process. The focus here is to ensure that adequate processes are used for each of the various types of systems engineering initiatives – "using the right tool for the job."

A small project which meets the criteria for *SEM Express* is typically straight-forward in nature and estimated to be less than 100 effort hours (including both systems development related and project management related hours). A large project, which meets the criteria for the full SEM, is typically complex in nature and is estimated at more than 400 effort hours. Projects that fall in the middle are considered medium projects, and will typically use a customized SEM for development of the system. Section 2.2.2 also offers guidance on customizing the SEM – giving guidance on which SEM templates to use, based on project work type.

The project manager has the discretion to use *SEM Express* for slightly larger projects if he/she feels the complexity is such that *SEM Express* is preferable.

If at any time the project manager feels he/she need to have more process guidance, he/she has the discretion to add processes and/or templates from the full SEM to meet the documentation/approval needs of the project. It is also acceptable to switch from *SEM Express* to a customized SEM mid-stream if the project warrants such a change, due to increased scope, inaccurate initial estimates, etc.

The following SEM Tailoring Matrix is designed to guide the project / development manager in selecting the relevant components of the Systems Engineering Methodology for use in their project.

This matrix is used to identify SEM templates and processes required for a given project size.

SEM Tailoring Matrix

NOTES:
1.) "If Applicable" means the template is required if the project has impact on that area, such as training, contract management, or infrastructure changes.
2.) It is assumed that if "master" documents exist for the system, those master documents will be updated and attached to the current SEM / SEM Express documents, with the new changes noted.

Template / Process	Document Reference	Small, Straight-Forward Project -*SEM Express*-	Medium Project -Customized SEM-	Large Project -SEM-	Guidance
EA Solution Assessment	SEM Touch Point (Solution Assessment Worksheet)	Not Applicable	Required if no existing EA Solution Assessment is on file with EA or if proposing changes to the one on file.	Required if no existing EA Solution Assessment is on file with EA or if proposing changes to the one on file.	Check with an Enterprise Architecture representative if unsure.
Maintenance Plan	SEM-0301	Integrated into *SEM Express* Initiation, Requirements, and Design Plan.	New plan required or updates to original plan, if available.	New plan required or updates to original plan, if available.	
Software Configuration Management Plan	SEM-0302	Integrated into *SEM Express* Initiation, Requirements, and Design Plan.	New plan required or updates to original plan, if available.	New plan required or updates to original plan, if available.	
Requirements Traceability Matrix	SEM-0401	Not Required. Integrated into *SEM Express* Initiation, Requirements, and Design Plan.	Required	Required	
Requirements Specification	SEM-0402	Integrated into *SEM Express* Initiation, Requirements, and Design Plan.	New specification required or updates to original specification, if available	Required	
Requirements Management Checklist	SEM-0403	Not Required	Not Required	Required	
Functional Design Document	SEM-0501	Integrated into *SEM Express* Initiation, Requirements, and Design Plan.	New design required or updates to original design, if available	New design required or updates to original design, if available	
Conversion Plan	SEM-0601	Integrated into *SEM Express* Initiation, Requirements, and Design Plan.	Required if converting existing data	Required if converting existing data	
Test Plan	SEM-0602	Integrated into S*EM Express* Initiation, Requirements, and Design Plan.	Required	Required	
Test Report	SEM-0603	Integrated into *SEM Express* Initiation, Requirements, and Design Plan.	Required	Required	

Template / Process	Document Reference	Small, Straight-Forward Project -*SEM Express*-	Medium Project -Customized SEM-	Large Project -SEM-	Guidance
System Design Document	SEM-0604	Integrated into *SEM Express* Initiation, Requirements, and Design Plan.	New design document required or updates to original design document, if available	Required	
System Design Checklist	SEM-0605	Not Required	Not Required	Required	
Software Testing Checklist	SEM-0606	Not Required	Not Required	Required	
Transition Plan	SEM-0701	Integrated into *SEM Express* Construction and Testing Plan.	If Applicable	Required	Required if new staffing or operational procedures are identified for operations staff, maintenance staff, or client staff
Installation Plan	SEM-0702	Integrated into *SEM Express* Construction and Testing Plan.	Required	Required	
Training Plan	SEM-0703	Integrated into *SEM Express* Construction and Testing Plan.	If Applicable	Required	Required if new staffing or training needs are identified
Training Checklist	SEM-0704	Not Required	If Applicable	Required	
Integration and System Testing Checklist	SEM-0801	Not Required	Required	Required	
Error Reporting and Tracking Checklist	SEM-0802	Not Required	Not Required	Required	
PreAcceptance Checklist	SEM-0803	Not Required	Not Required	Required	
Testing Package Checklist	SEM-0804	Integrated into *SEM Express* Construction and Testing Plan.	Required	Required	
User Acceptance Checklist	SEM-0805	Integrated into *SEM Express* Construction and Testing Plan.	Required	Required	Used for client signoff of the completed system
Structured Walkthrough process	Structured Walkthrough Process Guide	Required for both Initiation, Requirements, and Design Plan and Construction and Testing Plan	Required	Required	Structured Walkthroughs are required for all major deliverables
Stage Exit process	Stage Exit Process Guide	Required for each stage	Required for each stage	Required for each stage	

Template / Process	Document Reference	Small, Straight-Forward Project -*SEM Express*-	Medium Project -Customized SEM-	Large Project -SEM-	Guidance
Security Plan	SEM Tough Point (DIT-0170)	If Applicable	If Applicable	Required	
Infrastructure Services Request	SEM Touch Point (DIT 184)	If Applicable	If Applicable	Required	
Contracts and Procurement documents	SEM Touch Point (DIT-0153, DIT-0015A, DIT-0015B)	If Applicable	If Applicable	If Applicable	
Business Continuity Plan	SEM Touch Point	If Applicable	If Applicable	Required	

Section: **2.2.2 Work Type Definitions**

Description: There are 6 work types currently available within the SEM. Work types can be identified by either an alphabetic character designation or by a descriptive name. See *Exhibit 2.2.2 Work Type Selection Diagram* on page 28 to determine the work type that applies.

The work types documented in the SEM include the following:

(A) **Break/Fix**

The Break/Fix work type is used if there has been an interruption of a critical service to a client. An action is required and a solution must be put in place, even if the solution is temporary. The problem must be investigated to determine the root cause. The permanent solution to the problem may result in the initiation of another work type.

Examples:

Production abend
Loss of on-line production system
Incorrect or missing customer data
Hardware malfunction
Network lines down

Use your current process to handle this work type.

(B) **Enhancement/Maintenance**

The Enhancement/Maintenance work type applies to an application system modification involving process changes and/or data structure changes. There are no changes to hardware or software platforms. This work type assumes that programs will be changed or created.

Examples:

Changing a business rule
Adding or changing edit checks for validating data
Changing or creating a report layout
Changing or creating an update/display screen
Creating a new data structure
Changing the field length of a data structure

For this work type, the following SEM templates need to be completed:

Test Plan Template (SEM-0602)
Test Report Template (SEM-0603)
Transition Plan Template (SEM-0701)

For this work type the, following SEM templates will need to be revised, or created if they do not currently exist:

Requirements Specification Template (SEM-0402)
Functional Design Template (SEM-0501)
System Design Template (SEM-0604)
Security Plan Template (DIT-0170)

For this work type, the following SEM templates may be revised or created as needed:

Requirements Traceability Matrix (SEM-401)
PMM Charter (PMM-02) / Project Plan (PMM-03 or PMM-03 Exp)
Training Plan (SEM-0703)
Training Checklist (SEM-0704)

(C) **New Development**

The New Development work type applies to the development of a new application system.

Examples:

Developing a new Web application system
Developing a new Desktop application system

For this work type, the following SEM templates need to be completed:

Software Configuration Management Plan (SEM-302)
Security Plan (DIT-170)
Solution Assessments
PMM Charter (PMM-02) / Project Plan (PMM-03 or PMM-03 Exp)
Requirements Traceability Matrix (SEM-401)
Requirements Specification (SEM-402)
Functional Design Document (SEM-501)
System Design Document (SEM-604)
Test Plan (SEM-602)
Test Reports (SEM-603)
Transition Plan (SEM-701)
Training Plan (SEM-703)

For this work type the, following SEM templates will need to be revised, or created if they do not currently exist:

Business Continuity Planning documentation

For this work type, the following SEM templates may be revised or created as needed:

Maintenance Plan (SEM-301)
Procurement Docs (DIT-153, DIT-15a, DIT-15b)

Requirements Management Checklist (SEM-403)
Conversion Plan (SEM-601)
System Design Checklist (SEM-605)
Software Testing Checklist (SEM-606)
Training Checklist (SEM-704)

(D) Commercial Off The Shelf (**COTS) Implementation**
The COTS Implementation work type applies to the implementation of an existing application system. This includes the implementation of vendor provided "turn key" applications.

Examples:
COTS Application
ERNIE (DEQ)

For this work type, the following SEM templates need to be completed:

Security Plan (DIT-170)
Procurement Documents (DIT-153, DIT-15a, DIT-15b)
Solution Assessments
Requirements Traceability Matrix (SEM-401)
Requirements Specification (SEM-402)
Test Plan (SEM-602)
Test Reports (SEM-603)
Installation Plan (SEM-702)
Training Plan (SEM-703)

For this work type the, following SEM templates will need to be revised, or created if they do not currently exist:

Business Continuity Planning documentation

For this work type, the following SEM templates may be revised or created as needed:

Maintenance Plan (SEM-301)
Software Configuration Management Plan (SEM-302)
PMM Charter (PMM-02) / Project Plan (PMM-03)
Requirements Management Checklist (SEM-403)
Conversion Plan (SEM-601)
Software Testing Checklist (SEM-606)
Training Checklist (SEM-704)

(E) **Application Migration**
The Application Migration work type applies to the migration of an application to a new hardware or software platform. Conversion programs may be necessary. There are no changes to processes or data structures.

Examples:

Migrating an application from one data center to another data center
Porting an application from a PC UNIX platform to a Sun UNIX platform
Porting an application from a NT 4 server to Windows 2003 server cluster
Porting a database from Oracle to SQL Server

For this work type, the following SEM templates need to be completed:

Solution Assessments
Conversion Plan (SEM-0601)
Test Plan (SEM-0602)
Test Report (SEM-0603)

For this work type the, following SEM templates will need to be revised, or created if they do not currently exist:

Security Plan (DIT-170)
Business Continuity Planning documentation
Requirements Traceability Matrix (SEM-0401)
Requirements Specification (SEM-0402)

For this work type, the following SEM templates may be revised or created as needed:

Procurement Documents (DIT-153, DIT-15a, DIT-15b)
PMM Charter (PMM-02) / Project Plan (PMM-03 or PMM-03 Exp)
Requirements Management Checklist (SEM-0403)

Study

The Study work type is used to evaluate a client's business problem or opportunity which result in recommended solutions. These solutions may not always result in system related work.

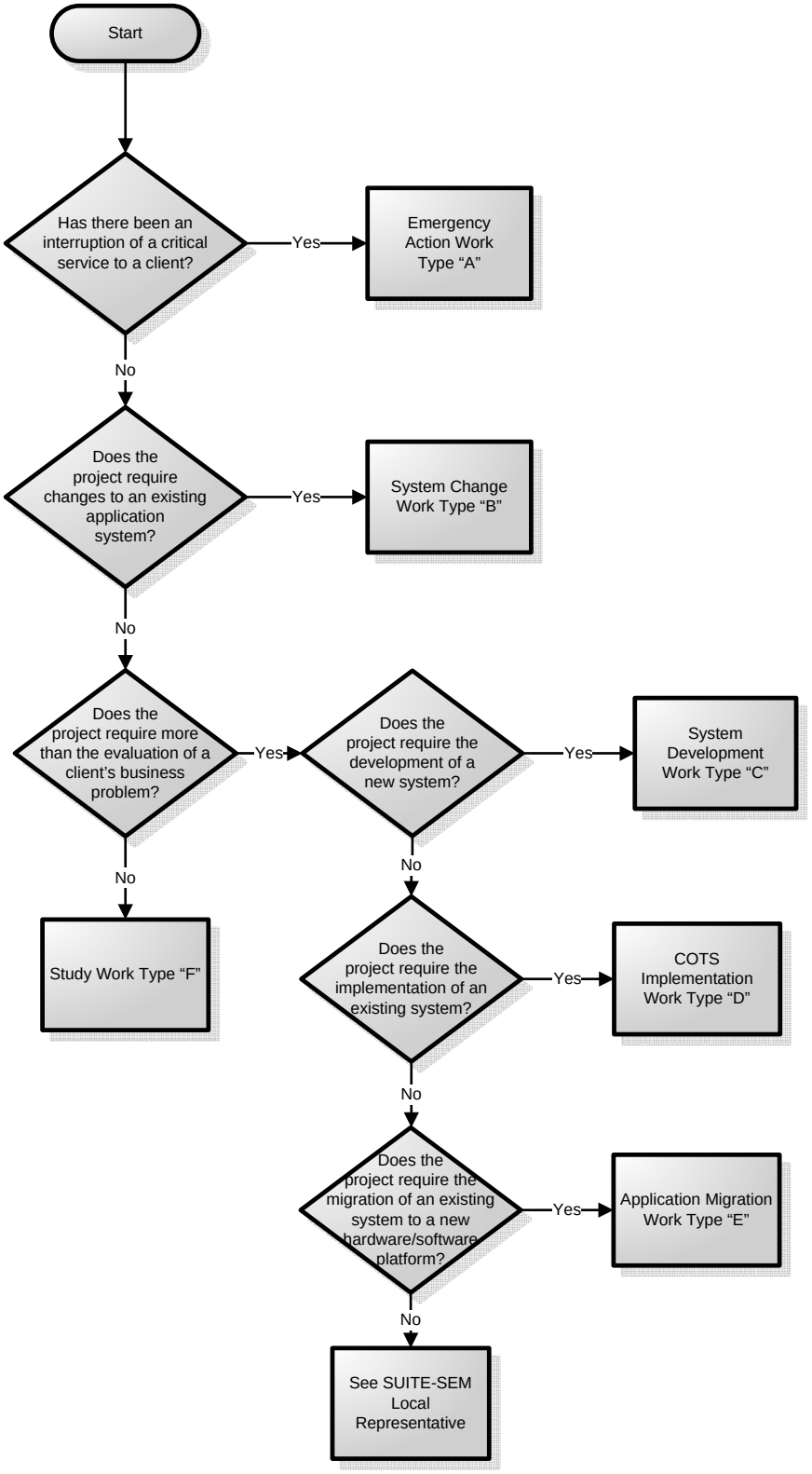
Examples:

- Providing a recommendation to decrease turn around time for accounts payable invoices
- Providing a recommendation for new application systems to replace old application systems.

For this work type, the following SEM templates need to be completed:

Requirements Specification (SEM-0402)
Solution Assessments

Exhibit 2.2.2 Work Type Selection Diagram



Section: **2.3 Development Techniques**

Description: This section provides descriptions of some development techniques that can be used with the SEM. The descriptions include high-level instructions on how to adapt the lifecycle stages to accommodate the development technique. The descriptions provided here are not intended to be a comprehensive list of development techniques.

Segmented Development:

NOTE: The term "segment" is used here to avoid confusion between project and production phases.

Segmented development is most often applied to large information systems engineering projects where the project requirements can be divided into functional segments. Each segment becomes a separate project and provides a useful subset of the total capabilities of the full product. This segmenting serves two purposes: to break a large development effort into manageable pieces for easier project management and control; and to provide intermediate work products that form the building blocks for the complete product.

The lifecycle processes and activities are applied to each segment. The overall system and software objectives are defined, the system architecture is selected for the overall project, and a Project Plan for development of the first segment is written and approved by the system owner.

Segments are delivered to the system owner for evaluation or actual operation. The results of the evaluation or operation are then used to refine the content of the next segment. The next segment provides additional capabilities. This process is repeated until the entire product has been developed. If significant problems are encountered with a segment, it may be necessary to reexamine and revise the project objectives, modify the system architecture, update the overall schedule, or change how the segments are divided.

Two major advantages of this approach are: the project manager can demonstrate concrete evidence that the final product will work as specified; and users will have access to, and use of, segments or functions prior to the delivery of the entire product.

Spiral Development:

Spiral development repeats the planning, requirements, and functional design stages in a succession of cycles in which the project's objectives are clarified, alternatives are defined, risks and constraints are identified, and a prototype is constructed. The prototype is evaluated and the next cycle is planned.

The project objectives, alternatives, constraints, and risks are refined based on this evaluation, then an improved prototype is constructed. This process of refinement and prototyping is repeated as many times as necessary to provide an incrementally firm foundation on which to proceed with the project.

The lifecycle activities for the Initiation and Planning, Requirements Definition, and Functional Design Stages are repeated in each cycle. Once the design is firm, the lifecycle stages for System Design, Construction, and Testing are followed to produce the final product.

Rapid Prototyping:

Rapid prototyping can be applied to any information systems development methodology (e.g., segmented, spiral.) Rapid prototyping is recommended for systems development that is based on a new technology or evolutionary requirements.

With the rapid prototyping technique, the most important and critical requirements are defined based on current knowledge and experience. A quick design addressing those requirements is prepared, and a prototype is coded and tested. The purpose of the prototype is to gain preliminary information about the total requirements and confidence in the correctness of the design approach. Characteristics needed in the final product, such as efficiency, maintainability, capacity, and adaptability might be ignored in the prototype.

The prototype is evaluated, preferably with extensive user participation, to refine the initial requirements and design. After confidence in the requirements and design approach is achieved, the final product is developed. The prototype might be discarded, or a portion of it used to develop the final product.

The normal documentation requirements are usually postponed with prototyping efforts. Typically, the project team, project stakeholders, and system owner agree that the prototype will be replaced with the actual product and required support documentation after proof of the model. The system that replaces the prototype should be developed using the lifecycle processes and activities.

Iterative Technique:

The iterative technique is normally used to develop products piece by piece. Once the system architecture and functional or conceptual design are defined and approved, system functionality can be divided into logically related pieces called "drivers."

In iterative fashion, the project team performs system design, code, unit test, and integration test activities for each driver, thereby delivering a working function of the product. These working functions or pieces of the product are designed to fit together as they are developed. This technique allows functions to be delivered incrementally for testing so that they can work in parallel with the project team. It also enables other functional areas, such as documentation and training, to begin performing their activities earlier and in a more parallel effort. In addition, the iterative technique enables progress to be visible earlier, and problems to be contained to a smaller scope.

With each iterative step of the development effort, the project team performs the lifecycle processes and activities.

Rapid Application Development: Rapid Application Development (RAD) is a method for developing systems incrementally and delivering working pieces every 3 to 4 months, rather than waiting until the entire project is constructed before implementation. Over the years, many information technology projects failed because by the time the implementation took place, the business had changed.

RAD employs a variety of automated design and development tools, including Computer-Aided Software Engineering (CASE), advanced generation languages, visual development, and graphical user interface (GUI) builders, which get prototypes up and running quickly. RAD focuses on personnel management and user involvement as much as on technology.

Joint Application Development: Joint Application Development (JAD) is a RAD concept that involves cooperation between the designer of a computer system and the end user to develop a system that meets the user's needs exactly. It complements other system analysis and design techniques by emphasizing participative development among system owners, users, designers, and builders. During JAD sessions for system design, the system designer will take on the role of facilitator for possibly several full-day workshops intended to address different design issues and deliverables.

Object-Oriented Development: Object-oriented development focuses on the design of components that mimic the real world. A component that adequately mimics the real world is much more likely to be used and reused. The approach emphasizes how a system operates, as opposed to analysis, which is concerned with what a system is capable of doing. One of the most important advantages in using an object-oriented approach is the ability to reuse components. Traditional practices surrounding development often mitigate against reuse. Short-term goals are stressed because today's milestones must be achieved before any thought can be given to milestones that may be months or years away.

Borrowed or reused software code is often code that has already been tested, and in the end, may translate into cost savings. Object-oriented development may make code reuse much easier but, the amount of actual reuse may still depend on the motivation of the project managers, designers and developers involved. Code reuse can also lead to faster development. Object-oriented systems are easier to maintain because their structures are inherently decoupled. This usually leads to fewer side effects when changes have to be made. In addition, object-oriented systems may be easier to adapt and scale (i.e., large systems can be created by assembling reusable subsystems).

Typically, the object-oriented process follows an evolutionary spiral that starts with customer communication, where the problem is defined. The technical work associated with the process follows the iterative path of analysis, design, construction, and testing. The fundamental core concepts in object-oriented design involve the elements of classes, objects, and attributes. Understanding the definition and relationships of these elements is crucial in the application of object-oriented technologies.

It is recommended that the following object-oriented issues be well understood in order to form a knowledge base for the analysis, design, testing, and implementation of systems using object-oriented techniques.

- What are the basic concepts and principles that are applicable to object-oriented thinking?
- How should object-oriented projects be planned and managed?
- What is object-oriented analysis and how do its various models enable a systems engineer to understand classes, their relationships and behavior?
- What is a “use case” and how can it be applied to analyze the requirements of a system?
- How do conventional and object-oriented approaches differ?
- What are the components of an object-oriented design model?
- How are “patterns” used in the creation of an object-oriented design?
- What are the basic concepts and principles that are applicable for testing of object-oriented systems?
- How do testing strategies and test case design methods change when an object-oriented system is considered?
- What technical metrics are available for assessing the quality of object-oriented systems?

Work Product: The work products described in the SEM will be the same for many of the development techniques and it is the responsibility of the project manager to adapt the work products accordingly and document adaptations in the Project Plan.

References: SOM Project Management Methodology: http://www.michigan.gov/projectmanagement See Section 3 – Project Planning Phase.

Section: **2.4 Commercial-Off-The-Shelf (COTS) Products Based Projects**

Description: There is a current trend in information systems development to make greater use of Commercial-Off-The-Shelf (COTS) products, that is, to buy a ready-made system from a software manufacturer rather than developing it in-house from scratch. This carries with it a sense of getting a system that can do the job at a reasonable cost, and getting new functions in subsequent releases over time. This practice is especially encouraged and sometimes mandated in government agencies. There can be many benefits in using COTS products including improving quality and performance, developing and delivering solutions more quickly, maintaining systems more cost effectively, and standardizing across the organization. The main characteristics of a COTS product are that it exists, is known to be proven, is available to the general public, and can be bought, leased, or licensed.

COTS and Open Systems: Many initiatives are under way in both private industry and government agencies including the State of Michigan to promote the use of an open systems approach, thereby anticipating even greater benefits than can be obtained from the use of COTS products alone. These initiatives are occurring because just buying COTS does not necessarily result in an "open" system. COTS products are not necessarily open, and they do not necessarily conform to any recognized interface standards. Therefore, it is possible that using a COTS product commits the user to proprietary interfaces and solutions that are not common with any other product, component, or system.

If the sole objective is the ability to capture new technology more cheaply, then the use of COTS products that are not open may satisfy requirements. However, considering that the average COTS component is upgraded every 6 to 12 months and new technology appears on the scene about every 18 to 24 months, any money that is saved by procuring a COTS product with proprietary interfaces may quickly be lost in maintenance as products and interfaces change.

In the midst of all this, interface standards provide a source of stability. Without such standards every change in the marketplace can impose an unanticipated and unpredictable change to systems that use products found in the marketplace.

COTS Planning Considerations: A COTS-based systems solution approach requires new and different investments including market research on available and emerging products and technologies, and COTS product evaluation and selection. The key to determining if the best solution is one which includes COTS products is to weigh the risks of straying from the three basic criteria - fully-defined, available to the public, and maintained according to group consensus - against what is to be gained over the

long term. An open systems approach requires investments in the following areas early in a project's lifecycle and on an ongoing basis:

- Market surveys to determine the availability of standards
- Selection of appropriate applicable standards
- Selection of standards-compliant implementations

These costs/activities are the necessary foundation for creating systems that serve current needs and yet can grow and advance as technology advances and the marketplace changes. On an ongoing basis, it is important for project teams to stay informed in this area, with particular focus on:

- When revisions to specific standards are scheduled for release
- What changes are proposed in the new revision
- When ballots on the revisions are going to occur
- Where the implementations are headed

Skills Considerations:

The depth of understanding and technical and management skills required on a project team are not necessarily diminished or decreased because of the use of COTS or open systems. The skills and understanding needed increase because of the potential complexity of integration issues, the need to seriously consider longer-term system evolution as part of initial development, and the need to make informed decisions about which products and standards are best.

Types of COTS Solutions:

COTS products can be applied to a spectrum of system solutions, including (but not limited to) the following:

- Neatly packaged solutions such as Microsoft Office that require no integration with other components.

- COTS products that support the information management domain, such as Oracle or SQL Server. These systems typically consist of both COTS products and customized components, with some "glue" code to enable them to work cooperatively.

- Systems comprised of a mix of COTS products and non-commercial products that provide large-scale functionality that is otherwise not available. Such systems typically require larger amounts of "glue" code to integrate the various components.

COTS Impact on the Project Lifecycle:

All systems engineering projects include planning, requirements definition, architecture definition, system design, code, test, and system integration activities. The use of COTS products has an impact on project lifecycle activities. The most fundamental change is that the system is now composed from building blocks that may or may not work cooperatively directly out of the box. The project team will require skilled engineering expertise to determine how to make a set of components work cooperatively - and at what cost.

This fundamental shift from development to composition causes numerous technical, enterprise, management, and business changes. Some of these changes are obvious, whereas others are quite subtle.

Requirements Definition

For a COTS-based system, the specified requirements must be sufficiently flexible to accommodate a variety of available commercial products and their evolution. To write such requirements, the author should be sufficiently familiar with the commercial marketplace to describe functional features for which actual commercial products exist.

There is a critical relationship among technology and product selection, requirement specification, and architecture definition. If the architecture is defined to fulfill the requirements and then the COTS product is selected, there may be only a few or no available products that fit within the chosen architecture. Pragmatically, three essential elements--requirements, architecture, and product selection--must be worked in parallel with constant trade-offs among them.

Adaptation/Integration

Assembling COTS products presents new challenges. Although COTS products are attempting to simulate the "plug and play" capability of the hardware world, in reality, they seldom plug into anything easily. Most products require some amount of adaptation and integration to work harmoniously with other commercial or custom components in the system. The typical solution is to adapt each COTS product through the use of "wrappers," "bridges," or other "glueware." It is important to note that adaptation does not imply modification of the COTS product. Adaptation can be a complex activity that requires technical expertise at the detailed system and specific COTS component levels. Adaptation and integration must take into account the interactions among custom components, COTS products, any non-developmental item components, any legacy code, and the architecture including infrastructure and middleware elements.

Testing

As the testing of COTS-based systems is considered, it must be determined what levels of testing are possible and needed. A COTS product is a "black box" and therefore changes the nature of testing. A system may use only a partial set of features of a given COTS product. In developing a test strategy and test plans, consideration should be given to issues such as should only the features used in the system be tested, and how does one test for failures in used features that may have abnormal behavior due to unknown dependencies between the used and unused features of a COTS product?

Maintenance

Maintenance also changes in very fundamental ways; it is no longer solely concerned with fixing existing functionality or incorporating new mission needs. Vendors update their COTS products on their schedules and at differing intervals. Also, a vendor may elect to eliminate, change, add, or combine features for a release. Updates to one COTS product, such as new file formats or naming convention changes, can have unforeseen consequences for other COTS products in the system. To further complicate maintenance, all COTS products will require continual attention to license expirations and changes. All of these events routinely occur. All of these activities may (and typically do) start well before an organization installs the system or a major upgrade. Pragmatically, the distinction between development and maintenance all but disappears.

Adapting the SEM for COTS Projects:

All systems engineering projects have a project lifecycle, require project management activities such as project planning, requirements definition, project tracking, software configuration management, and quality assurance; and produce deliverables such as project plans, requirements specifications, software configuration management plans, and test plans. At the same time, each project, whether COTS or traditional, can vary in scope, duration, technology used or operating platform. The SEM can be used as the project lifecycle for COTS-based projects as well as for traditional systems development and maintenance projects where all of the code is developed “in-house.”

The key to using the SEM effectively for COTS projects lies in adapting the lifecycle stages and deliverables to best suit the individual needs and characteristics of each particular project. See *Exhibit 2.4-1, Example of SEM Adapted for COTS Projects*, for an example of how to adapt the SEM for a COTS project. Stages should be combined as appropriate if, for example, a project will have a relatively small scope, and/or short duration, and/or will use known technology. On the other hand, the traditional number of stages may be appropriate for large projects with new technology and long duration.

Deliverables may be added to, or deleted from the standard list prescribed by the SEM (see *Exhibit 2.0-1, SEM Overview Diagram* on page 13). For COTS-based projects, the lifecycles stages will typically include "evaluation," "selection," "customization," and "integration," and the project deliverables will typically include documents such as "Products to be Evaluated," and "COTS Solution Recommendations." See *Exhibit 2.4-1 Example of SEM Adapted for COTS Projects.*

Documenting Deviations:

The adaptation (or deltas) from the standard SEM prescribed stages and deliverables are known as deviations. These deviations should be documented with an explanation in the project plan. Deviations from prescribed project deliverables should be documented with an explanation, and a statement, which describes how project risk is not elevated if a prescribed deliverable will not be produced.

Resources:

The following references are from the features section of the Carnegie Mellon University Software Engineering Institute Website and were used in the preparation of this chapter:

- Software Technology Review: COTS and Open Systems
- Monthly Features: The Opportunities and Complexities of Applying COTS
- Monthly Features: Discussion with Members of the SEI COTS-Based Systems Initiative
- Software Technology Review: Components-based Software Development/COTS integration

Exhibit 2.4-1 Example of SEM Adapted for COTS Projects

SEM Stages	SEM/COTS Project Stages	Deliverables	
		SEM/COTS Project Planned Deliverables	Adaptation vs. SEM Deliverables
Initiation & Planning	Initiation & Planning	• Security Plan (PMM) • Project Plan (includes WBS) (PMM)	• Prototype instead of Concept Document (PMM) • Software Configuration Management Plan moved to Requirements Definition
Requirements Definition	Requirements Definition	• Functional Requirements Document • Business Continuity Plan • Products to be Evaluated • Software Configuration Mgmt Plan • Preliminary Data Requirements • Requirements Traceability Matrix	• The System and Acceptance Test Requirements will be developed in the COTS Evaluation and Selection Stage
Functional Design System Design	Evaluation and Selection	• COTS Solution/Recommendation • System Architecture • System/Acceptance Test Requirements • Conversion Plan	• Functional Design and System Design stages are combined • System Architecture document replaces System Design document • Logical Model, Physical Model, Construction Specifications, Coding Practices not applicable
Construction Testing	Customization, Testing	• Solution Baseline • Training Plan • User Documentation • System Maint. Documentation • Transition Plan • Integration Test Checklist • Test Report • Installation Plan • Pre-Acceptance Checklist	• The Construction and Testing stages are combined • Integration portion of the Test Plan not required
Implementation	Implementation	• User Training Materials • User Acceptance Checklist • Operational System	• No Deviations

Section: **2.5 Quality Reviews**

Description: This section describes the quality review and assurance mechanisms that are used with the SEM. The purpose of quality reviews is to assure that the established information systems development and project management processes and procedures are being followed effectively, and that exposures and risks to the current project plan are identified and addressed. The quality reviews facilitate the early detection of problems that could affect the reliability, maintainability, availability, integrity, safety, security, or usability of the software product. The quality reviews enhance the quality of the end work products and deliverables of a project.

Work products are subject to quality reviews. Quality reviews are conducted as Peer Reviews, Structured Walkthroughs (SWT) and Stage Exits. The quality review used depends on the work product being reviewed, the point of time within the project stage, and the purpose of the review.

Review Process: **Peer Review**

A peer review is an informal review of information systems engineering work products, including documentation, which can be conducted at any time at the discretion of the work product developer. These informal reviews are performed by the developer's "peers"-- frequently other developers working on the same project. Informal reviews can be held with relatively little preparation and follow up activity. Review comments are informally collected and the product developer determines which comments require future action. Some of the work products prepared are considered interim work products as they feed into a major deliverable or into another stage. Interim work products are ideal candidates for peer review; however, all work products benefit from peer reviews.

Responsibility

Team Members

Review Process: **Structured Walkthrough**

The Structured Walkthrough (SWT) is a more formal review and is prescribed by the SEM for all project deliverables. SWTs are used to find and remove errors from work products early and efficiently, and to develop a better understanding of defects that might be prevented. They are very effective in identifying design flaws, errors in analysis or requirements definition, and validating the accuracy and completeness of deliverable work products.

SWTs are conducted during all stages of the project lifecycle. They are used during the development of work products identified as deliverables for each stage (see *Exhibit 2.0-1 SEM Overview Diagram* on page 13), such as requirements, specifications, design, code, test cases (scripts), and documentation. SWTs are used after the work products have been completed to verify the correctness and the quality of the finished product. They should be scheduled in the work

breakdown structure developed for the project plan, where, in practice, they are sometimes referred to generically as reviews. SWTs should also be scheduled to review small, meaningful pieces of work. The progress made in each lifecycle stage should determine the frequency of the walkthroughs; however, they may be conducted multiple times on a work product to ensure that it is free of defects.

SWTs can be conducted at various times in the development process, in various formats, with various levels of formality, and with different types of participants. They typically require some advance planning activities, a formal procedure for collecting comments, specific roles and responsibilities for participants, and have prescribed follow-up action and reporting procedures. Frequently reviewers include people outside of the developer's immediate peer group.

Responsibility
Project Manager, Team Members, Work Product Author, Reviewers

Work Products
A SWT Meeting Record (DIT-0187) is available for the reviewers to record errors found prior to the walkthrough session, and for the scribe to record information discussed during the walkthrough. Upon completion, the presenter or author of the work product compiles a SWT Management Summary Report (DIT-0188) and a copy is placed in the Project File.

Reference
The State of Michigan guidance document titled *Structured Walkthrough Process Guide* provides a procedure and sample forms that can be used for SWTs. This document is available on the MDIT SUITE website.

Review Process:

Stage Exit
The Stage Exit is a process for ensuring a project meets the project standards and milestones identified in the project plan. The Stage Exit is conducted by the project manager with the project stakeholders (e.g., system owner and the following points of contact: user, quality assurance, security, architecture and standards, project manager's manager, and platform). It is a high-level evaluation of all work products developed in a lifecycle stage. It is assumed that each deliverable has undergone several peer reviews and/or SWTs as appropriate prior to the Stage Exit process. The Stage Exit focuses on the satisfaction of all requirements for the stage of the lifecycle, rather than the specific content of each deliverable.

The goal of a Stage Exit is to secure the approval of designated key individuals to continue with the project and to move forward into the next lifecycle stage. The approval is a sign-off of the deliverables for the current stage of development including the updated project plan. It indicates that all qualifications (issues and concerns) have been closed or have an acceptable plan for resolution. At a Stage Exit meeting, the project manager communicates the positions of the key personnel, along with qualifications raised during the stage exit process, and the

action plan for resolution to the project team, stakeholders, and other interested meeting participants. The Stage Exit meeting is documented in summary form. Only one Stage Exit for each stage should be necessary to obtain approval assuming all deliverables have been accepted as identified in the project plan.

A Stage Exit is an effective project management tool that is required for all projects regardless of size. For small projects, stages can be combined and addressed during one Stage Exit.

Responsibility
Project Manager.

Work Products
A Stage Exit Position Response form (DIT-0189) is completed by each approver. A summary of the Stage Exit meeting is prepared by the project manager or a designee and distributed to the meeting attendees. The summary identifies any issues and action items needed to obtain concurrence prior to proceeding to the next lifecycle stage.

Reference: The MDIT guidance document titled *Stage Exit Process Guide* provides a procedure and sample report form that can be used for Stage Exits. This document is available on the MDIT SUITE website.

Chapter: **3.0 Initiation and Planning Stage**

Description: This is the first stage in the lifecycle of an information systems engineering project. Project Management Methodology (PMM) and System Engineering Methodology are tightly integrated. It is imperative that the project manager ensures participation of the business client in the creation of the PMM documents.

The majority of the outputs produced during this stage are Project Management Methodology (PMM) documents, such as the Business Case, the Project Charter, the Project Plan and the Quality Management Plan. The SEM Software Configuration Management Plan and the Maintenance Plan are initiated during this stage.

Although the following italicized paragraphs focus on PMM processes, a solid project management foundation is necessary for SEM deliverables and subsequent stages.

Project planning applies to all projects regardless of their size. Planning involves selecting the strategies, policies, programs, and procedures for achieving the objectives and goals of the project. Planning is deciding, in advance, what to do, how to do it, when to do it, where to do it, and who is going to do it.

The requirements identified in project related materials, (e.g., a business case document, are the primary input to the Project Plan.) The level of detail will vary depending on project size. The preparation of the Project Plan and related materials involves several critical planning issues such as the identification of preliminary requirements; staff, schedule, and cost estimates; the technical and managerial approaches that will be used; and the assessment of potential risks associated with the project. This information forms the foundation for all subsequent planning activities.

During this stage, the system owner and users are interviewed to: identify their business needs and expectations for the product; gain a common understanding of the task assignment; and determine how the project supports the State of Michigan's long-range information resource management plans. The system owner is the enterprise unit that is funding the project, and users are the State of Michigan employees and contractors who will use the product.

In this stage, the project team should be focused on identifying what the project will automate, and whether developing an IT solution makes sense from business, cost, and technical perspectives. If the project is feasible, then time, cost, and resource estimates must be formulated for the project, and risk factors must be assessed. It is important for the project team to work closely with representatives from all functional areas that will be involved in providing resources, information, or support services for the project (see Touch Points). The information gathered in this stage is used to plan and manage the project throughout its lifecycle.

This stage involves development of a Software Configuration Management Plan (SEM) to track and control work products and a Quality Management Plan (PMM) to assure the production and operation of high quality products on schedule, within budget, and within the constraints specified by the system owner and user.

Input: The following items provide input to this stage:

- Requirements identified in project related materials, (e.g., a business case)
- Related project initiation materials

High-Level Activities: The remainder of this chapter is divided into sections that describe specific high-level activities performed during this stage. These activities represent the minimum requirements for a large information systems engineering effort. Notes are provided, as applicable, to assist in customizing these lifecycle stage requirements to accommodate different size efforts. The high-level activities are presented in the sections listed below.

3.1 Develop Software Configuration Management Plan
3.2 Develop Maintenance Plan

Touch Points: The following touch points are involved in the Initiation and Planning Stage:

- Contracts and Procurement
  - Assignment of a Contract Liaison if procuring goods or services
  - Completion on DIT-0153 Bid Information Sheet if procuring goods or services

- Enterprise Architecture (EA)
  - Review relevant EA materials (e.g., roadmaps, solution patterns)
  - Develop EA Solution Assessment for each alternative (refer to Appendix C for assistance in developing the EA Solution Assessment.

- Security
  - Notify your Security Liaison of project initiation
  - Review MDIT and Agency Security Policies
  - Initiate Security Plan, including Data Classification and System Criticality sections

- Other
  - Initiate Business Continuity Planning process (DMB has a website for this purpose.)

Output:

Several work products are developed during this stage. The work products listed below are the minimum requirements for a large project. Deviations in the content and delivery of these work products are determined by the size and complexity of a project. Explanations of the work products are provided under the applicable activities described either in the remainder of this chapter or in the PMM.

SEM Templates:

- Software Configuration Management Plan (*initial*)
- Maintenance Plan (*initial*)

PMM Templates:

- Business Case
- Concept Document (i.e., Feasibility Study)
- Project Charter
- Project Plan (includes Quality Management Plan)
- Security Plan (*initial*)

Other Outputs:

- Enterprise Architecture (EA) Solution Assessment for each potential solution option (*initial*)
- Business Continuity Plan (*initial*)

A diagram showing the work products associated with each SEM stage is provided in *Exhibit 3.0-1, SEM Overview – Initiation and Planning Stage Highlighted.* The activities for this stage are emphasized in bold.

Review Process:

Quality reviews are necessary during this stage to validate the project and associated work products. The activities that are appropriate for quality reviews are identified in this chapter and chapter 2.0, Lifecycle Model. The time and resources needed to conduct the quality reviews should be reflected in the project resources, schedule, and work breakdown structure.

Structured Walkthrough (SWT)

Requirements for a peer review or a more formal structured walkthrough are documented under *Review Process* at the end of each Task, Subtask, or Activity section in this stage. The State of Michigan guide titled *Structured Walkthrough Process Guide* provides a procedure and sample forms that can be used for SWTs. This document is available on the MDIT SUITE website.

Stage Exit

Schedule a Stage Exit as the last activity of the Initiation and Planning Stage to enable the project approvers to review project deliverables and provide a concur/non-concur position to the project manager. The State of Michigan guide titled *Stage Exit Process Guide* provides a procedure and sample report form that can be used for stage exits. This document is available on the MDIT SUITE website.

References: Chapter 2.0, Lifecycle Model, *Quality Reviews*, provides an overview of the Quality Reviews to be conducted on a project.

Bibliography: The following materials were referenced in the preparation of this chapter.

1. Carnegie Mellon University, Software Engineering Institute, *Capability Maturity Model: Guidelines for Improving the Software Process*, Addison Wesley Longman, Inc., 1994.

2. Project Management Institute, *A Guide to the Project Management Body of Knowledge*, Pennsylvania, 1996.

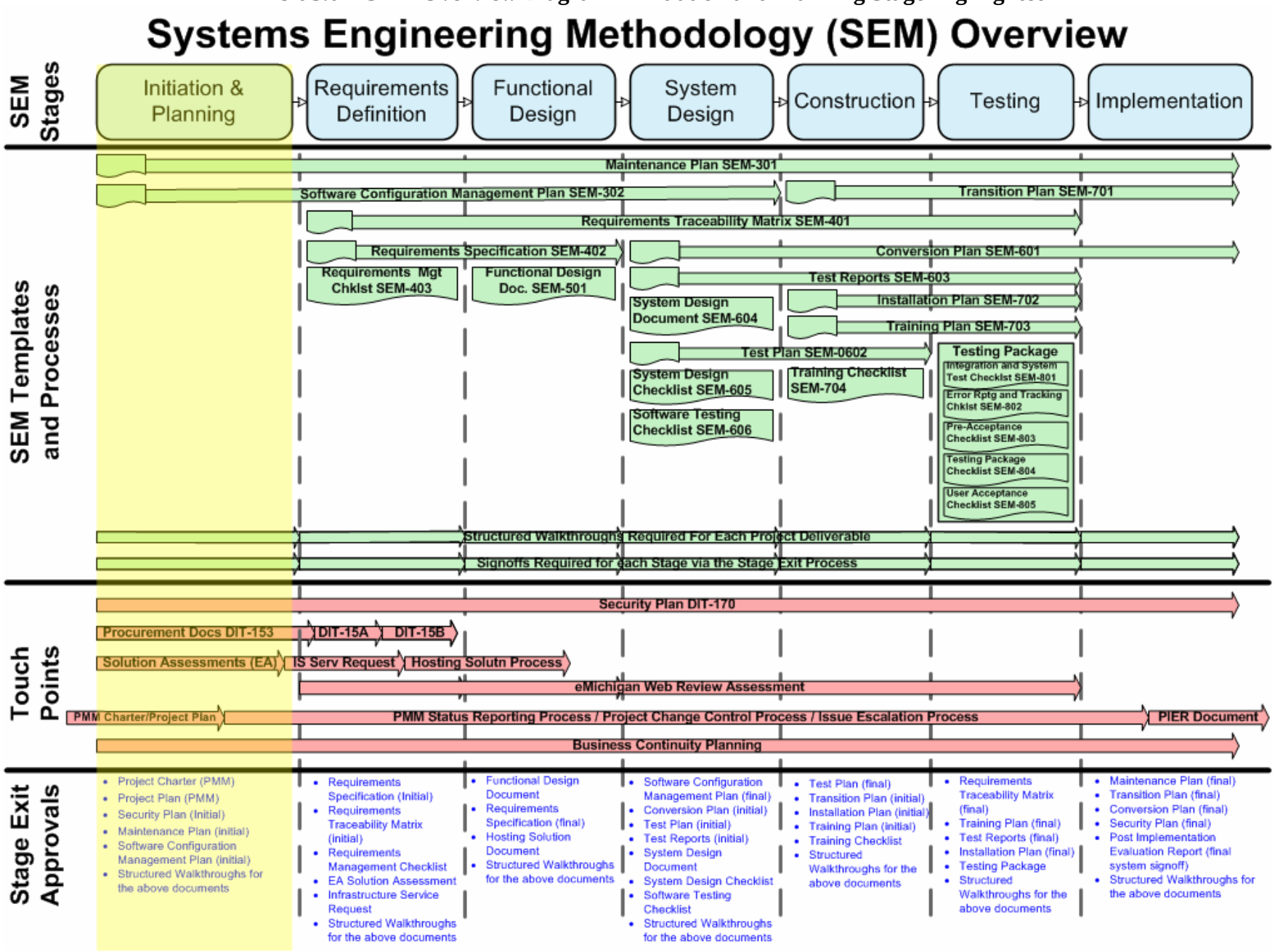
3. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Developing Software Lifecycle Processes*, IEEE Std 1074-1991, New York, 1992.

4. U.S. Department of Energy, *Departmental Information Systems Engineering: Volume 1, Information Systems Engineering Lifecycle,* September 2000.

5. U.S. Department of Energy, *DOE/NV Software Management Plan*, Nevada Operations Office, May 1991.

6. U.S. Department of Energy, *Software Management Guide*, DOE/AD-0028, 1992.

Exhibit 3.0-1 SEM Overview Diagram – Initiation and Planning Stage Highlighted



Activity: **3.1 Develop Software Configuration Management Plan**

Responsibility: Project Manager or Software Configuration Manager

Description: Software configuration management (SCM) is a standard and consistent set of processes used to control version changes to work products and other artifacts during all stages of the project lifecycle. The goals of configuration management are to identify the configuration of the product (i.e., work products and their descriptions) at given points in time, to systematically control changes to the configuration, and to maintain the integrity and traceability of the configuration throughout the lifecycle.

Software configuration management includes the following activities:

- An SCM plan is prepared for each project according to a documented process and is coordinated with affected groups and individuals.
- A documented and approved software configuration management plan is used as the basis for performing the software configuration management activities.
- Authority for managing the project's baselines is established (e.g., a Software Configuration Control Board - SCCB).
- An SCM library system is established as a repository for the baselines.
- Work products are identified and placed under configuration control.
- Change requests and problem reports for all items/units are initiated, recorded, reviewed, approved, and tracked according to a documented process.
- Changes to baselines are controlled according to a documented process.
- Products from the baseline library are created and their release is controlled according to a documented procedure.
- Status of items/units is recorded according to a documented process.
- Standard reports documenting the SCM activities and the contents of the baseline are developed and made available to affected groups and individuals.
- Baseline audits are conducted according to a documented process.

The work products placed under SCM include the products that are delivered to the customer (e.g., the requirements documents and the source code) and the items that are identified with or required to create these products (e.g., the compiler). A baseline library is established containing the baselines of the configuration items as they are developed. Changes to baselines and the release of products built from the baseline library are systematically controlled via the change control and configuration auditing functions of configuration management.

The Software Configuration Manager (or the individual assigned configuration responsibilities) is responsible for routine evaluation of the product. The Software Configuration Manager controls changes that are introduced into the systems product environment. The Software Configuration Manager is responsible for the processes necessary to correct faults in the environment and product. The Software Configuration Manager is not responsible for any overall project deadlines or management issues.

Work Product: An SCM plan that defines the configuration management policies and procedures is required for each project. The plan is developed early in the lifecycle to ensure the control of changes as soon as the project requirements are approved and baselined. In this stage, the plan addresses activities that are platform independent, such as identifying the items that will be placed under configuration management. As the project progresses through the lifecycle stages, the plan is expanded to reflect platform specific activities.

The SCM plan addresses the following types of responsibilities and activities:

- Defining the required configuration management policy and procedures.
- Maintaining all documents in an easily accessible central library.
- Receiving unit test and integrated builds and related documentation from the developer.
- Performing version control procedures on unit and integrated builds received from the developer.
- Packaging a tested unit or integrated build for return to developer or production as required.
- Shipping an approved unit or integrated build to production as required.
- Maintaining an archive of project-related correspondence between members of the project team.
- Overseeing the release and subsequent distribution of configuration items.

Provide enough information in the plan so that compliance can be monitored by means of project records. Whenever feasible, acquire automated SCM tools to check compliance with project standards, the validity and consistency of product design, requirements, and system performance.

Based on the complexity of the project and the anticipated volume of changes, a Software Configuration Management Plan can be developed for a specific project, an existing plan can be modified to suit the requirements of a project, or a plan can be developed to manage all of the projects supporting a particular system owner's organization. Place a copy of the SCM Plan in the Project File.

Review Process: Conduct structured walkthroughs to validate that the configuration management approach, the configuration identification, change control, status accounting, and auditing procedures are appropriate for the project.

Resources: The MDIT SUITE website contains a sample SCM plan and template.

The Institute of Electrical and Electronics Engineers (IEEE) Standard for Software Configuration Management Plans (Std 828-1990) provided guidance on developing these plans.

Activity: **3.2 Develop Maintenance Plan**

Responsibility: Project Manager

Description: The purpose of the Maintenance Plan is to determine the scope of the maintenance effort, identify the process and tools, quantify the maintenance effort (personnel and resources), and identify anticipated maintenance requirements. The Maintenance Plan needs to define the maintenance process and its boundaries or scope. The maintenance process beginning point should be defined (e.g., receipt of a change request or planned COTS version upgrade) and the ending action should be defined (e.g., delivery and sign-off of a product). The process is a natural outgrowth of many of the software configuration management procedures. A description of the overall flow of work within the maintenance process should be included. The maintenance process can be tailored to the type of maintenance being performed and can be divided in several different ways. This can include different processes for corrections or enhancements or small or large changes.

The maintenance requirements need to be identified and quantified (sized) in the Maintenance Plan to determine the future maintenance load for the organization. The following issues should be considered when defining the requirements.

- Expected external or regulatory changes to the product
- Expected internal changes to support new requirements
- Requirements deferred from current project to later release
- Wish-list of new functions and features
- Expected upgrades for performance, adaptability, or connectivity
- New lines of business that need to be supported
- New technologies that need to be incorporated

The requirements for the maintenance staff also need to be established. At this stage, the maintenance plan should begin to address the following:

- Number of maintainers, their job descriptions, and required skill levels
- Experience level of the maintenance staff
- Documented maintenance processes at the systems and program levels
- Actual methods used by development staff
- Tools used to support the maintenance process
- Current work load and estimates of future needs

Description, continued:

An important part of the maintenance plan is an analysis of the hardware and software most appropriate to support the maintenance organization's needs. The development, maintenance, and test platforms should be defined and differences between the environments described. Tools sets that enhance productivity should be identified and provided. Tools should be accessible to all who need them, and sufficient training provided so that their use will be well understood.

Although all systems need maintenance, there comes a time when maintenance is no longer technically or fiscally viable. Issues such as resources, funds, and priorities may dictate that a system should be replaced rather than changed. The maintenance plan should identify the criteria that indicate the product is ready for retirement or replacement, such as the failure rate, age of code, and incompatibility with current technology.

Work Product: Initiate development of the Maintenance Plan. Place a copy of the draft Maintenance Plan in the Project File. As part of the Implementation Stage, once system installation and acceptance are complete, determine if the Maintenance Plan needs to be revised, if so, update and finalize the Maintenance Plan. Place a copy of the initial Maintenance Plan in the Project File.

Review Process: Conduct a structured walkthrough to ensure that the Maintenance Plan accurately reflects the necessary information.

The Maintenance Plan is formally reviewed during the Stage Exit process.

Reference: Refer to the forthcoming MDIT *Systems Maintenance Guide*, for more information on maintaining the product.

Resource: A template for the Maintenance Plan is available on the MDIT SUITE website.

Chapter: **4.0 Requirements Definition Stage**

Description: The primary goal of this stage is to develop a basis of mutual understanding between the business owner/users and the project team about the requirements for the project. The result of this understanding is an approved Requirements Specification that becomes the initial baseline for product design and a reference for determining whether the completed product performs as the system owner requested and expected. All system requirements, (e.g., software, hardware, performance, functional, infrastructure, etc.) should be included.

This stage involves analysis of the business owner/users' business processes and needs, translation of those processes and needs into formal requirements, and planning the testing activities to validate the performance of the product.

Input: The following work products provide input to this stage:

SEM Templates:

- Software Configuration Management Plan
- Maintenance Plan

PMM Templates:

- Business Case
- Concept Document (i.e., Feasibility Study)
- Project Charter (Statement of business objectives)
- Project Plan (includes Quality Management Plan)
- Security Plan

Other Inputs:

- Enterprise Architecture (EA) Solution Assessment for each potential solution option

High-Level Activities: The remainder of this chapter is divided into sections that describe specific high-level activities performed during this stage.

4.1 Requirements Management
4.2 Select Requirements Analysis Technique
4.3 Define System Requirements
4.4 Compile and Document System Requirements
4.5 Develop System Test Requirements
4.6 Develop Acceptance Test Requirements
4.7 Establish Functional Baseline

Touch Points: The following touch points are involved in the Requirements Definition Stage:

- Contracts and Procurement
  - Completion of DIT-0015a, if procuring commodities (e.g., servers, software)
  - Completion of DIT-0015b (including Statement of Work and Requirements Traceability Matrix), if procuring services (e.g., project management, application developers)
  - Utilize the services of the assigned Contract Liaison, if procuring services

- E-Michigan
  - Web review assessment by E-Michigan's webmaster to ensure ADA compliance and Michigan.gov look and feel standards. Contact E-Michigan for more information on obtaining this review.

- Enterprise Architecture (EA)
  - Use relevant EA materials (e.g., roadmaps, solution patterns) while developing Technical Requirements
  - Revise/complete EA Solution Assessment for each alternative
  - Refer to the EA TechTalk portal site

- Infrastructure Services
  - When EA Solution is complete and approved, prepare Infrastructure Services Request (ISR), and begin Hosting Solution document

- Security
  - Review MDIT and Agency security policies
  - Review State and Federal laws and regulations
  - Begin Infrastructure/Network and Data Flow Diagram

Output: Several work products are developed during this stage. The work products listed below are the minimum requirements for a large systems project. Deviations in the content and delivery of these work products are determined by the size and complexity of a project. Explanations of the work products are provided under the applicable activities described in the remainder of this chapter.

SEM Templates:

- Requirements Specification *(initial)*
- Requirements Management Checklist
- Requirements Traceability Matrix *(initial)*
- Maintenance Plan *(revised)*
- Software Configuration Management Plan *(revised)*

PMM Templates:

- Project Plan *(revised)*
- Security Plan *(revised)*

Other Outputs:

- Application Hosting document *(initial)*
- Business Continuity Plan *(revised)*
- EA Solution Assessment *(final)*
- Infrastructure Services Request (*final*)

A diagram showing the work products associated with each high-level activity is provided in *Exhibit 4.0-1, SEM Overview Diagram – Requirements Definition Stage Highlighted.*

Once the requirements are baselined, changes must be managed through the established change process, which may include a change control board.

Review Process: Quality reviews are necessary during this stage to validate the product and associated work products. The activities that are appropriate for quality reviews are identified in this chapter and Chapter 2.0, Lifecycle Model.

Structured Walkthrough (SWT)
Requirements for a peer review or a more formal structured walkthrough are documented under *Review Process* at the end of each Task, Subtask, or Activity section in this stage. The State of Michigan guide titled *Structured Walkthrough Process Guide* provides a procedure and sample forms that can be used for SWTs. This document is available on the MDIT SUITE website.

Stage Exit
Schedule a Stage Exit as the last activity of the Requirements Definition Stage to enable the project approvers to review project deliverables and provide a concur/non-concur position to the project manager. The State of Michigan guide titled *Stage Exit Process Guide* provides a procedure and sample report form that can be used for stage exits. This document is available on the MDIT SUITE website.

References: Chapter 2.0, Lifecycle Model, Q*uality Review*, provides an overview of the Quality Reviews to be conducted on a project.

Bibliography: The following materials were referenced in the preparation of this chapter.

1. Carnegie Mellon University, Software Engineering Institute, *Capability Maturity Model: Guidelines for Improving the Software Process*, Addison Wesley Longman, Inc., 1994.

2. *Software Engineering Handbook*, Chapter 4, Software Requirements Analysis.

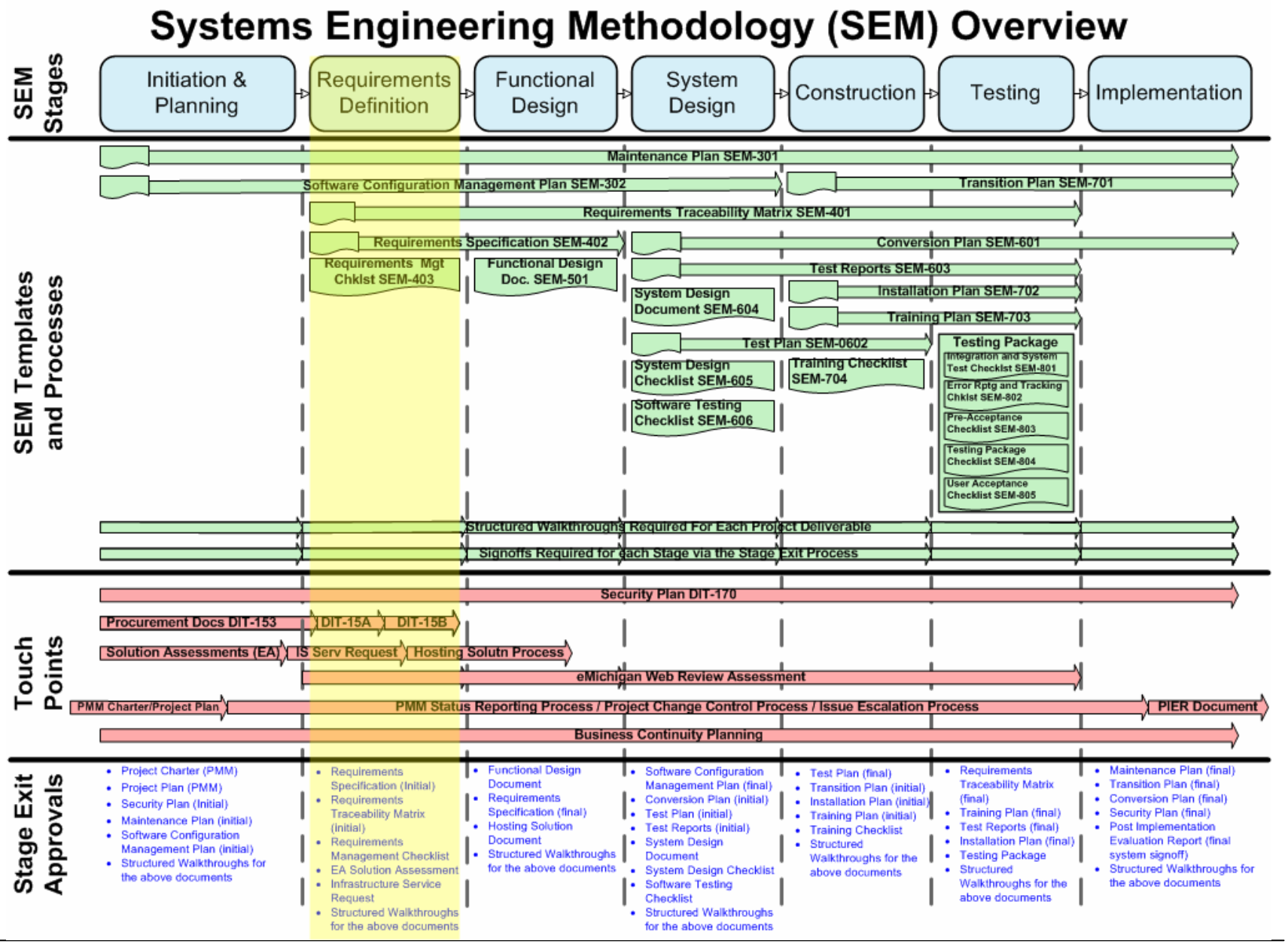
3. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Guide to Software Requirements Specifications*, ANSI/IEEE Std 830-1998, New York, 1998.

4. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Developing Software Lifecycle Processes*, IEEE Std 1074-1991, New York, 1992.

5. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Software Verification and Validation Plans*, ANSI/IEEE Std 1012-1998, New York, 1998.

6. U.S. Department of Energy, Nuclear Weapons Complex, Quality Managers, Software Quality Assurance Subcommittee, *Guidelines for Software Requirements Management*, July 1998.

Exhibit 4.0-1 SEM Overview Diagram – Requirements Definition Stage Highlighted



Activity: **4.1 Requirements Management**

Responsibility: Project Manager / Business Area Manager

Description: Requirements management is essentially a process composed of gathering, organizing, prioritizing, and documenting requirements; verifying that requirements have been captured in the product, and managing changes to requirements.

Gathering, organizing, prioritizing and documenting requirements is an interactive communication process and working relationship between stakeholders and the project team to discover, define, refine, and record a precise representation of the product requirements.

Requirements management documents the needs, expectations, and understanding of the product to be delivered and provides a framework for identifying, planning, scheduling, costing, verifying, tracing, testing, evaluating, changing, and renegotiating requirements to satisfy stakeholder needs and expectations of the project. When requirements are initially gathered, some or all will be planned for the current project (e.g., initial release). The requirements for the project are documented in the Requirements Specification document. As the project progresses, more requirements may be identified and managed through a change control process. As part of requirements management, the project manager must track requirements that are accepted for the current project and those that will be planned for subsequent releases.

Each requirement in the Requirements Specification document should be uniquely identified in a Requirements Traceability Matrix. The Requirements Traceability Matrix is a requirements management tool that ensures requirements are traced and verified through the various lifecycle stages – especially design, testing, and implementation. Requirements must be traceable from external sources such as the customer, to derived system-level requirements, to specific hardware/software product requirements.

Work Products: A substantial amount of information that is used for requirements management in later stages in the systems engineering process is gathered in the Requirements Definition Stage. The Requirements Management Checklist should be used to ensure that all requirements management activities are performed. The Requirements Traceability Matrix is a work product that is created during the Requirements Definition Stage and used to verify and validate that requirements are met and the product remains within scope. Refer to each task for information on applicable work products.

Resources: Templates for the Requirements Traceability Matrix, Requirements Management Checklist, and Requirements Specification document are provided on the MDIT SUITE website.

Templates for the Software Change Request Form and the Software Change Control Log are provided on the MDIT SUITE website.

Carnegie Mellon University, Software Engineering Institute, *Capability Maturity Model: Guidelines for Improving the Software Process*, Addison Wesley Longman, Inc., 1994.

Tasks: The following task is involved in requirements management.

4.1.1 Develop Requirements Traceability Matrix

Task: **4.1.1 Develop Requirements Traceability Matrix**

Description: A requirements traceability matrix is a tool used to trace project lifecycle activities and work products to the project requirements. The matrix establishes a thread that traces requirements from identification through implementation.

Every project requirement must be traceable back to a specific project objective(s) described in the Project Charter. This traceability assures that the product will meet all of the project objectives and will not include inappropriate or extraneous functionality.

All work products developed during the design, code, and testing processes in subsequent lifecycle stages must be traced back to the project requirements described in the Requirements Specification. This traceability assures that the product will satisfy all of the requirements and remain within the project scope.

It is also important to know the source of each requirement, so that the requirements can be verified as necessary, accurate, and complete. Meeting conference records, user survey responses, and business documents are typical sources for project requirements.

Work Product: Develop a matrix to trace the requirements back to the project objectives identified in the Project Charter and forward through the remainder of the project lifecycle stages. Place a copy of the matrix in the Project File. Expand the matrix in each stage to show traceability of work products to the requirements and vice versa. The requirements traceability matrix contains descriptions for each item in the matrix.

Review Process: Conduct a structured walkthrough of the Requirements Traceability Matrix to ensure that all requirements have been accurately captured.

Sample Traceability Matrix: One method for tracing requirements is a threading matrix that groups requirements by project objectives. Under each project objective, the source of the requirement, the unique requirement identification number, and the lifecycle activities are listed in columns along the top and the project requirements in rows along the left side. As the project progresses through the lifecycle stages, a reference to each requirement is entered in the cell corresponding to the appropriate lifecycle activity. A template example for the Requirements Traceability Matrix is provided on the MDIT SUITE website.

Resource: A template for the Requirements Traceability Matrix is provided on the MDIT SUITE website.

Activity: **4.2 Select Requirements Analysis Technique**

Responsibility: Project Manager/Team

Description: A requirements analysis technique is the set of data collection and analysis techniques (e.g., Joint Application Design [JAD], user interviews, and rapid prototyping) combined with the lifecycle requirements standards (e.g., tracing the requirements through all lifecycle activities) that are used to identify the project requirements and to define exactly what the product must do to meet the system owner/users' needs and expectations. When appropriate, the technique must include methods for collecting data about users at more than one geographic location and with different levels and types of needs.

The requirements analysis technique should be in harmony with the type, size, and scope of the project; the number, location, and technical expertise of the users; and the anticipated level of involvement of the users in the data collection and analysis processes. The technique should ensure that the functionality, performance expectations, and constraints of the project are accurately identified from the system owner/users' perspective. The technique should facilitate the analysis of requirements for their potential impact on existing operations and business practices, future maintenance activities, and the ability to support the system owner's long-range information resource management plans.

It is advantageous to select a technique that can be repeated for similar projects. This allows the project team and the system owner/users to become familiar and comfortable with the technique.

Discuss the analysis technique with the business owner and users to make sure they understand the process being used, their role and responsibilities in the process, and the expected format of the output (e.g., how the requirements will be organized and described).

Work Product: Create a description of the analysis technique and share it with all members of the project team, business owner, and users.

Review Process: Conduct a structured walkthrough to verify that the requirements analysis technique is appropriate for the scope and objectives of the project. A structured walkthrough is not needed when the technique has been used successfully on similar projects for the same system owner/user environment.

Activity: **4.3 Define System Requirements**

Responsibility: Project Manager/Team

Description: Use the project scope, objectives, and high-level requirements as the basis for defining the system requirements. The questions used to define the business objectives may be helpful in developing the system requirements. The goals for defining system requirements are to identify what functions are to be performed on what data, to produce what results, at what location, and for whom.

The requirements must focus on the products that are needed and the functions that are to be performed. Avoid incorporating design issues and specifications in the requirements. One of the most difficult tasks is to determine the difference between "what" is required and "how to" accomplish what is required. Generally, a requirement specifies an externally visible function or attribute of a system (i.e., "what".) A design describes a particular instance of how that visible function or attribute can be achieved (i.e., "how to").

Requirements should be specified as completely and thoroughly as possible. The requirements must support the business owner's business needs, information resource management long-range plans, and the enterprise (SOM-DIT) and Department-Agency missions. When requirements are being defined, it is not sufficient to state only the requirements for the problems that will be solved; all of the requirements for the project must be captured.

Attributes: Each requirement must be stated as a unique objective with the following attributes. The existence of these attributes must be verified prior to the delivery of the Requirements Specification later in the Requirements Definition Stage.

- Necessary - Absolute requirements that are to be verified are identified by "must" or "shall". Goals or intended functionality are indicated by "will".
- Correct - Each requirement is an accurate description of a feature or process of the product.
- Unambiguous - The statement of each requirement denotes only one interpretation.
- Complete - Each requirement describes one result that must be achieved by the product. The requirement should not describe the means of obtaining the result.

- Consistent - Individual requirements are not in conflict with other requirements.
- Verifiable (testable) - Each requirement is stated in concrete terms and measurable quantities. A process should exist to validate that the product (when developed) will satisfy the set of requirements.
- Modifiable - The structure and style of the requirements are such that any necessary changes to the requirements can be made easily, completely, and consistently.
- Traceable - The origin of each requirement is clear and can be tracked in future development activities and tests.

Identification System: The creation of a standard identification system for all requirements is required in order to facilitate configuration control, requirements traceability, and testing activities. The identification system must provide a unique designator for each requirement. For example, the identification system can classify the requirements by type (e.g., functional, input, or computer security). Within each type classification, the requirements can be assigned a sequential number. Select an identification system that is appropriate for the scope of the project.

Changes: As the project evolves, the requirements may change or expand to reflect modifications in the users' business plans, design considerations and constraints, advances in technology, and increased insight into user business processes. A formal change control process must be used to identify, control, track, and report proposed and approved changes. Approved changes in the requirements must be incorporated into the Requirements Specification in such a way as to provide an accurate and complete audit trail of the changes. This change control process should be an integral part of the project's Software Configuration Management Plan.

Tasks: The following tasks are involved in defining system requirements.

4.3.1 Define Functional Requirements
4.3.2 Define Input and Output Requirements
4.3.3 Define Performance Requirements
4.3.4 Define User Interface Requirements
4.3.5 Define System Interface Requirements
4.3.6 Define Communication Requirements
4.3.7 Define Computer Security and Access Requirements
4.3.8 Define Backup and Recovery Requirements
4.3.9 Define Preliminary Implementation Requirements

Task: **4.3.1 Define Functional Requirements**

Description: Functional requirements define what the product must do to support the system owner's business functions and objectives. The functional requirements should answer the following questions.

- How are inputs transformed into outputs?
- Who initiates and receives specific information?
- What information must be available for each function to be performed?

Identify requirements for all functions whether they are to be automated or manual. Describe the automated and manual inputs, processing, outputs, and conditions for all functions. Include a description of the standard data tables and data or records that will be shared with other applications. Identify the forms, reports, source documents, and inputs/outputs that the product will process or produce to help define the functional requirements.

Develop a functional model to depict each process that needs to be included. The goal of the functional model is to represent a complete top-down picture of the product.

Use flow diagrams to provide a hierarchical and sequential view of the system owner's business functions and the flow of information through the processes.

Work Product: Maintain a record of all functional requirements. Save for incorporation into the Requirements Specification.

Optional Work Product: Consider developing an optional work product that defines how the final product will operate to support the system owner organization's business functions and objectives. This user-oriented requirements manual would identify processes in a narrative form from the user's perspective and would include requirements for all functions whether they are to be automated or manual. A functional description can be developed to depict each process that will be provided. The goal is to present a complete top-down picture of the product.

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the functional requirements.

Task: **4.3.2 Define Input and Output Requirements**

Description: Describe all manual and automated input requirements for the product such as data entry from source documents and data extracts from other applications; include where the inputs are obtained.

Describe all output requirements for the product such as printed reports, display screens, and files; include who or what is to receive the output.

Data requirements identify the data elements and logical data groupings that will be stored and processed by the product. The identification and grouping of data begins during the Requirements Definition Stage and is expanded in subsequent stages as more information about the data is known.

Work Product: Maintain a record of all input and output requirements. Save for incorporation into the Requirements Specification.

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the input and output requirements.

Task: **4.3.3 Define Performance Requirements**

Description: Performance requirements define how the product must function (e.g., hours of operation, response times, and throughput under various load conditions). The information gathered in defining the project objectives can translate into very specific performance requirements; (e.g., if work performed for an organization is mission essential to the Department, the hours of operation and throughput will be critical to meeting the mission). Also, MDIT and agency policy can dictate specific availability and response times.

Work Product: Maintain a record of all performance requirements. Save for incorporation into the System Requirements Document.

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the performance requirements.

Task: **4.3.4 Define User Interface Requirements**

Description: The user interface requirements should describe how the user will access and interact with the product, and how information will flow between the user and the product.

Interface Issues: A standard set of user interface requirements may be established for the system owner organization. If not, work with the system owner and users to develop a set of user interface requirements that can be used for all automated products for the system owner's organization. A standard set of user interface requirements will simplify the design and code processes, and ensure that all automated products have a similar look and feel to the users. When other constraints (such as a required interface with another application) do not permit the use of existing user interface standards, an attempt should be made to keep the user interface requirements as close as possible to the existing standard.

The following are some of the issues that should be considered when trying to identify user interface requirements.

- The users' requirements for screen elements, navigation, and help information.
- The standards issued by the SOM, MDIT and agencies that apply to user interfaces.
- The classification of the users who will access and use the product.
- The range of work that the users will be performing with the product.

Define the user interface requirements by identifying and understanding what is most important to the user, not what is most convenient for the project team.

Work Product: Maintain a record of all user interface requirements. Save for incorporation into the Requirements Specification.

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the user interface requirements.

Task: **4.3.5 Define System Interface Requirements**

Description: The hardware and software interface requirements must specify hardware and software interfaces required to support the development, operation, and maintenance of the product.

The following information should be considered when defining the hardware and software interface requirements.

- Business owner's and users' IT environment.
- Existing or planned software that will provide data to or accept data from the product.
- Other organizations or users having or needing access to the product.
- Purpose or mission of interfacing software.
- Common users, data elements, reports, and sources for forms/events/outputs.
- Timing considerations that will influence sharing of data, direction of data exchange, and security constraints.
- Development constraints such as the operating system, database management system, language compiler, tools, utilities, and network protocol drivers.
- Standardized system architecture defined by hardware and software configurations for the affected organizations, programmatic offices, sites, or telecommunications operations.

Work Product: Maintain a record of all system interface requirements. Save for incorporation into the Requirements Specification.

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the system interface requirements.

Task: **4.3.6 Define Communication Requirements**

Description: The communication requirements define connectivity and access requirements within and between user locations and between other groups and applications.

The following factors should be considered when defining communication requirements.

- Communication needs of the user and customer organizations.
- User organization's existing and planned communications environment (e.g., telecommunications; LANs, WANs, and dial-up).
- Projected changes to the current communication architecture, such as the connection of additional local and remote sites.
- Limitations placed on communications by existing hardware and software including:
    - user systems
    - applications that will interface with the product
    - organizations that will interface with the product
- SOM, MDIT and agency standards that define communication requirements and limitations.
- Future changes that may occur during the project.

Work Product: Maintain a record of all communication requirements. Save for incorporation into the Requirements Specification.

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability accuracy, and completeness of the communications requirements.

Task: **4.3.7 Define Computer Security and Access Requirements**

Description: Develop the computer security requirements in conjunction with the system owner's Office of Enterprise Security Liaison and Agency Security Office, and other stakeholders who provide competent input in the information system security area. This involvement affords early determination of classifications and levels of access protection required for the product.

If a product under development processes sensitive personal information, appropriate safeguards must be established to protect the information from accidental disclosure. Refer to the Department of Management and Budget website for guidance on the Privacy Act.

Implement applicable security procedures to assure data integrity and protection from unauthorized disclosure, particularly during development efforts. The organization that owns the data defines the data classification. The project team must be aware of all the types of data and of any classified or proprietary algorithms used in the product.

Procedure: Use the following procedure to determine computer security requirements.

1. Identify the types of data that will be processed by the product.
2. Determine preliminary data protection requirements.
3. Coordinate with the owner of the host platform to identify existing supporting computer security controls, if applicable.
4. Incorporate security requirements into the Requirements Specification.

Sample Access Control Questions: The following list provides sample questions that can be used to help define the access controls for the product.

- What access restrictions are placed on the users by their organization or programmatic office?
- What are the audit and other checking needs for the product?
- What separation of duties, supervisory functions related to control, operating environment requirements, or other functions will impact the product?
- What measures will be used to monitor and maintain the integrity of the product and the data from the user's viewpoint?

Work Product: Maintain a record of all security and access requirements. Save for incorporation into the Requirements Specification.

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the computer security and access requirements.

Resources:

- MDIT Office of Enterprise Security Resource Guide
- MDIT Office of Enterprise Security website

Task: **4.3.8 Define Backup and Recovery Requirements**

Description: Review any applicable existing Disaster Recovery agreements, Continuity of Operations agreements, and Service Level Agreements. Amend if needed to incorporate specific requirements for data backup, recovery and operational status for the product. Ensure that any mission essential systems are included in the Business Continuity Plan.

Work Product: Maintain a record of all data backup, recovery and operational start up requirements. If a product is determined to be mission essential, a Business Continuity Plan must be developed. If the product is not mission essential, a Continuity of Operations Statement within the plan is required. Place a copy of the Business Continuity Plan in the Project File.

Review Process: Conduct structured walkthroughs as needed to assure the necessity, testability, accuracy, and completeness of the backup and recovery requirements.

Resource: A template for the Business Continuity Plan is available on the DMB website.

Task: **4.3.9 Define Preliminary Implementation Requirements**

Description: Describe the requirements anticipated for implementing the product (e.g., user production cycle). The high-level implementation requirements are identified early in the lifecycle to support decisions that need to be made for the information systems engineering approach. The implementation requirements are expanded into a full implementation approach during the design stages.

The following paragraphs provide highlights of some of the implementation requirements that need to be considered.

Operating Environment: Identify any capacity restrictions on the existing hardware or software that needs to be addressed and identify any hardware or software that needs to be acquired (e.g., communication hardware, file servers, commercial off-the-shelf software, network interface cards, and LAN utilities).

Acquisition: If hardware or software must be acquired, identify the necessary acquisition activities. These activities include preparing specifications, estimating costs, scheduling procurement activities, selection, installation, and testing.

Conversion: Identify requirements for converting data from an existing or external application to the new product. Consider requirements for data entry, data protection, computer time, conversion programs, personnel, and other resources that will be needed. Also identify the requirements for the conversion of software, if necessary. Implementing a new application may involve converting software from one environment to another or modifying software to interface with other applications. Include requirements for testing the conversion process and validating that it was successfully accomplished.

Installation: Identify the installation requirements for any new hardware, operating system, or software. For hardware installations, consider environmental factors such as air conditioning, power supply, and security requirements. For software installations, consider proprietary software such as database management systems. For application software, consider the installation of the application's programs, parallel operation of the old and new applications, or the cutover from a test to a production environment. Hardware and software installation must be coordinated with the work cycles of the user organization to create a minimum of disruption, and to assure that data are available as needed. Installation must be scheduled to assure that, when data conversion is necessary, the needed data are protected.

Training: Identify the specific training needs for various categories of users and administrators. Also identify training requirements for personnel (e.g., agency staff, Michigan businesses, and Michigan citizens) time, computer time, training facilities, and training database(s).

Documentation: Identify requirements for the development and distribution of operational documentation for support personnel and user documentation. Operational documentation may include job control procedures and listings, operational instructions, system administration responsibilities, archiving procedures, and error recovery. User documentation includes the users manual, step-by-step instructions, online documentation, and online help facilities.

Work Product: Maintain a record of all implementation requirements. Save for incorporation into the Requirements Specification.

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the implementation requirements.

Activity: **4.4 Compile and Document System Requirements**

Responsibility: Project Manager/Team

Description: Compile the requirements gathered during the requirements analysis process in preparation for the development and delivery of the draft Requirements Specification. The following steps should be performed as part of the requirements compilation activity.

- Select and use a standard format for describing the requirements. Ensure compliance with Enterprise Architecture and any site specific standards.
- Present the logical and physical requirements without dictating a physical design or technical solutions.
- Write the requirements in non-technical language that can be fully understood by the system owner and users.
- Organize the requirements into meaningful groupings (e.g., all security-related requirements or all requirements for generating reports).
- Develop a numbering scheme for the unique identification of each requirement.
- Select a method for: (1) tracing the requirements back to the sources of information used in deriving the requirements (e.g., specific system owner/user project objectives); and (2) threading requirements through all subsequent lifecycle activities (e.g., testing).

Activity: **4.5 Develop System Test Requirements**

Responsibility: Project Manager

Description: The System Test Requirements section of the Requirements Specification document is a narrative and tabular description of the test activities planned for the project during development or enhancement. System Test Requirements should establish the testing necessary to validate that the project requirements have been met and that the deliverables are at an acceptable level in accordance with existing standards. System Test Requirements also ensure that a systematic approach to testing is established and that the testing is adequate to verify the functionality of the product.

System Test Requirements includes the resources, project team responsibilities, and management techniques needed to plan, develop, and implement the testing activities that will occur throughout the lifecycle. If individuals outside of the project team perform system and acceptance testing, the plan includes the responsibilities and relationships of external test groups.

In this stage, System Test Requirements are written at a high level and focus on identifying test techniques and test phases. Detailed information about test products (i.e., test plans, test procedures, and test reports) is added to the System Test Requirements as the project progresses through subsequent lifecycle stages.

Development of the System Test Requirements is the responsibility of the project manager. If a test group outside the project team will be involved in any test phase, the project manager must coordinate the System Test Requirements with each test group.

The System Test Requirements must be reviewed and approved by the system owner prior to conducting any tests.

Notes: For small projects, formal System Test Requirements may not be necessary; however, a test approach and testing are required and must be documented.

Typically, System Test Requirements cover all test phases including integration, system, and acceptance.

Work Product: When the System Test Requirements are completed, they should contain the following information:

- Describe the occurrence and timing of the test phases in the lifecycle and the entrance and exit criteria for each test phase.

- Specify the test products at each test phase. Describe the types and scope of the testing activities to be performed on each component of the application and the group who is responsible to produce them.
- Map what requirements are verified in what test phase.
- Establish the criteria for evaluating the test results of each test phase.
- Make an initial determination of the resources necessary to accomplish the testing.
- Identify the appropriate person or group to conduct each type of testing activity.
- Outline the test environment (hardware, software, test tools, and data) needed to conduct the tests.
- Develop a preliminary schedule for executing the test activities.

Review Process: Conduct structured walkthroughs to assure the System Test Requirements adequately describe all testing activities, test schedules, test products, test responsibilities, the testing methodology, and the required resources.

Resources: The Requirements Specification template includes a section for System Test Requirements. The Requirements Specification Template is available on the MDIT SUITE website.

Tasks: Preparation of the System Test Requirements involves the following tasks.

4.5.1 Identify Test Techniques
4.5.2 Identify Test Phases
4.5.3 Identify Test Environment Requirements

Task: **4.5.1 Identify Test Techniques**

Description: The System Test Requirements should specify the testing techniques planned for the project including the types of tests required, test documents, test methods, and test data collection. Each test from integration through acceptance testing is specified in terms of entrance and exit criteria and the expected level of involvement from the project team, test group, and other functional areas.

Integration tests with appropriate data must be developed to exercise and validate all specified application requirements, functions, and objectives. System and acceptance tests validate that the integrated system meets the requirements.

Each type of test must use controlled computer generated or live data as specified. The test data must be prepared to include values that will verify the functional capabilities of the test component, identify its limitations and deficiencies (if any), exercise its capabilities, and verify that the component performs its intended function as required.

If pilot testing or a phased implementation is required for the product, the System Test Requirements should include such requirements. In the case of an implementation involving phased releases, the plan should include the requirements for regression testing of the complete application as new elements are introduced.

For each type of test conducted, the test results are compared with the expected results. Discrepancies are identified and any problems resolved. Retesting is required to verify that the problem solution eliminates the problem and does not introduce new errors. The final test results are accompanied by a completed test results/error log form. This form is completed by the individual(s) responsible for testing and attached to the documents that certify the completion of each type of test.

Task: **4.5.2 Identify Test Phases**

Description: The product should be tested in sequential phases: integration, system, and acceptance. Some projects may require additional types of tests. The test phases are described below.

Integration Test Phase: Integration testing is an orderly progression of testing in which software elements, hardware elements, or both are combined and tested to evaluate the interaction between them. Each program/module must be tested. Integration testing is required to validate that groups of related programs, when combined to establish an integrated functional module of code, interface properly, and perform the functions for which they were designed. Examine the source program/module statements to ensure that the program logic meets the requirements of the design and that the application satisfies an explicit functional requirement. This test phase is performed by the project team.

System Test Phase: The system test phase tests the integrated hardware and software to verify that the product meets its specified requirements and operates successfully on the host platform. This test phase is required to validate, when the entire product is loaded onto the host platform, that the proper initialization is performed; decision branching paths are appropriate; and all functions are performed as specified in the Requirements Specification. System testing validates that the product produces the required outputs and interfaces properly with other systems with which the product gives or receives data; that transaction response times meet user expectations; and machine resource allocation and utilization are within expected norms. This test phase can be performed by the project team or by an independent test group with support from the project team.

Acceptance Test Phase: Acceptance testing is conducted to determine whether a product satisfies its acceptance criteria and to enable the system owner's organization to determine whether to accept the product. The acceptance test is required to validate that the system, its related documentation and tools, satisfy all of the specified requirements and objectives of the system owner's organization, State of Michigan standards, the Requirements Specification, and the design criteria. Acceptance testing will include tests of all intrasystem interfaces; and the use of all manuals, documentation, procedures, and controls. This test phase can be performed by the project team with system owner and user observers or by system owner and user representatives with support from the project team.

Task: **4.5.3 Identify Test Environment Requirements**

Description: The System Test Requirements should outline what is needed to perform testing activities throughout the project lifecycle including personnel, hardware, software, space, and other environmental requirements. As much testing as possible should be performed on the same equipment that will be used for the production system. In many cases, this information is not fully known until the System Design Stage.

The following are some of the considerations for test environment requirements.

- Evaluate automated testing tools for the following:
    - Generation of test scripts
    - Creation of result and error repositories
    - Consideration of each tool's benefits and costs
    - Use of simulators
- Determine local area network, wide area network, and metropolitan area network testing environment(s), as needed
- Determine test lab, data generation, and error correction support
- Identify Beta test sites

Activity: **4.6 Develop Acceptance Test Requirements**

Responsibility: Project Team

Description: The Acceptance Test Requirements section of Requirements Specification document is a description of the test activities planned for project acceptance. The Acceptance Test Requirements should establish the testing necessary to validate that the project requirements have been met and that the deliverables are at an acceptable level in accordance with existing standards. The Acceptance Test Requirements also assures that a systematic approach to acceptance testing is established and that the testing is adequate to verify the functionality of the product.

The complete set of system requirements and acceptance criteria form the basis for determining the overall approach to acceptance testing and the specific testing and examination methods. Features of the installation site and the system affect how the acceptance testing will be done. Unique arrangements may be necessary when the product cannot be completely installed and executed in a live environment. Multiple configurations may have to be distributed at several installation sites.

When a new system is a replacement for one already in use, the acceptance test must assure the integrity of the users' business operations while placing the replacement into operation. For example, the old system and the new system are used in parallel until complete functionality has been verified. In some cases, the acceptance process may take several months to assure that a complete business or accounting cycle has occurred. This concern will influence the approach to acceptance testing.

Work Product: Acceptance testing must be documented carefully with traceability of test cases to the requirements and acceptance criteria established by the system owner. At a minimum, the Acceptance Test Requirements should address the following requirements.

- Identification of the personnel involved in the acceptance test process and their testing responsibilities. If individuals outside of the project team perform acceptance testing, include the responsibilities and relationships of external test groups.
- Traceability of test designs and cases to requirements.
- The objectives and constraints for each test.
- Complete test cases and test procedures including inputs and expected outputs for each test case.

- Descriptions of error reporting, analysis, and resolution.
- Location(s) where testing will occur, the testing approach, type of facilities, and tester training.
- Acquisition of special purpose testing equipment, tools, and software.
- Resources and cost estimation to accomplish testing.

Review Process: Conduct structured walkthroughs to assure the draft Acceptance Test Requirements adequately describes all testing activities, test schedules, test products, test responsibilities, the testing methodology, and the required resources.

Resources: The Requirements Specification template includes a section for Acceptance Test Requirements. The Requirements Specification Template is available on the MDIT SUITE website.

Activity: **4.7 Establish Functional Baseline**

Responsibility: Project Manager/Team

Description: The functional baseline, sometimes called a system requirements baseline, is the main technical work product of the Requirements Definition Stage. The system requirements are baselined after the system owner's formal approval of the Requirements Specification. Once the requirements are baselined, any changes to the requirements must be managed under change control procedures established in the Software Configuration Management Plan. Approved changes must be incorporated into the Requirements Specification.

Work Product: Prepare the final Requirements Specification and submit to the system owner and users for their review and approval. The approved Requirements Specification is the official agreement and authorization to use the requirements for the product design. Approval implies that the requirements are understood, complete, accurate, and ready to be used as the basis for the subsequent lifecycle stages.

It is important for the system owner/users to understand that changes to the approved Requirements Specification affect the project scope and therefore can change the project cost, resources, or schedule. It is the responsibility of the project manager and project team to identify system owner/user requested changes that would result in a change of project scope; evaluate the potential impact to the project costs, resources, or schedule; and notify the system owner of the project planning revisions that will be required to accommodate their change requests. The Requirements Specifications are validated using the Requirements Management Checklist.

Place a copy of the approved Requirements Specification and the Requirements Management Checklist in the Project File.

Review Process: The Requirements Specification should be reviewed by the system owner and users. After making the changes needed to resolve problems found during the review, the functional baseline is formally established upon receipt of the system owner's approval.

Chapter: **5.0 Functional Design Stage**

Description: The functional design process maps the "what to do" of the Requirements Specification into the "how to do it" of the design specifications. During this stage, the overall structure of the product is defined from a functional viewpoint. The functional design describes the logical system flow, data organization, system inputs and outputs, processing rules, and operational characteristics of the product from the user's point of view. The functional design is not concerned with the software or hardware that will support the operation of the product or the physical organization of the data or the programs that will accept the input data, execute the processing rules, and produce the required output.

The focus is on the functions and structure of the components that comprise the product. The goal of this stage is to define and document the functions of the product to the extent necessary to obtain the system owner and users understanding and approval and to the level of detail necessary to build the system design.

Prototyping of system functions can be helpful in communicating the design specifications to the system owner and users. Prototypes can be used to simulate one function, a module, or the entire product. Prototyping is also useful in the transition from the functional design to the system design.

Input: The following work products provide input to this stage:

SEM Templates:

- Maintenance Plan
- Requirements Specification
- Requirements Traceability Matrix
- Software Configuration Management Plan

PMM Templates:

- Project Plan
- Quality Management Plan
- Security Plan

Other Inputs:

- Business Continuity Plan
- MDIT Hosting Solution Document

High-Level Activities: The remainder of this chapter is divided into sections that describe specific high-level activities performed during this stage. These activities represent the minimum requirements for a large information systems engineering effort. Notes

are provided, as applicable, to assist in customizing these lifecycle stage requirements to accommodate different sizes of systems engineering efforts.

The high-level activities are presented in the sections listed below.

5.1 Determine System Structure
5.2 Design Content of System Inputs and Outputs
5.3 Design User Interface
5.4 Design System Interfaces
5.5 Design System Security Controls
5.6 Build Logical Model
5.7 Build Data Model
5.8 Develop Functional Design
5.9 Select System Architecture

Touch Points: The following touch points are involved in the Functional Design Stage:

- Contracts and Procurement
  - Contract liaison involvement in the process, if contract issues arise
- E-Michigan
  - Continue to work with E-Michigan's webmaster, as appropriate, to ensure ADA compliance and Michigan.gov look and feel standards.
- Infrastructure Services
  - Review and complete Hosting Solution document
- Security
  - Review MDIT and Agency security policies
  - Review State and Federal laws and regulations
  - Review existing or propose new security controls
  - Conduct preliminary risk analysis
  - Revise Infrastructure/Network and Data Flow Diagram

Output: Several work products are developed during this stage. The work products listed below are the minimum requirements for a large systems project. Deviations in the context and delivery of these work products are determined by the size and complexity of a project. Explanations of the work products are provided under the applicable activities described in the remainder of this chapter.

SEM Templates:

- Functional Design Document (*final*)
- Maintenance Plan (*revised*)

- Requirements Specification (*final*)
- Requirements Traceability Matrix *(revised)*
- Software Configuration Management Plan (*revised*)

PMM Templates:

- Project Plan (*revised*)
- Security Plan (*revised*)

Other Outputs:

- Business Continuity Plan *(revised)*
- Data Dictionary (*final*)
- MDIT Hosting Solution Document (*final*)

A diagram showing the work products associated with each high-level activity is provided in *Exhibit 5.0-1, SEM Overview Diagram – Functional Design Stage Highlighted*

Review the Project Plan for accuracy and completeness of all Functional Design Stage activities and make any changes needed to update the information.

Review Process: Quality reviews are necessary during this stage to validate the product and associated work products. The activities that are appropriate for quality reviews are identified in this chapter and Chapter 2.0, Lifecycle Model. In addition, a Preliminary Design Review will be conducted. This review is an important milestone in the design process. The time and resources needed to conduct the walkthroughs and Functional Design Review should be reflected in the project resources, schedule, and work breakdown structure.

Structured Walkthrough (SWT)
Requirements for a peer review or a more formal structured walkthrough are documented under *Review Process* at the end of each Task, Subtask, or Activity section in this stage. The State of Michigan guide titled *Structured Walkthrough Process Guide* provides a procedure and sample forms that can be used for SWTs. This document is available on the MDIT SUITE website.

Stage Exit
Schedule a Stage Exit as the last activity of the Functional Design Stage to enable the project approvers to review project deliverables and provide a concur/non-concur position to the project manager. The State of Michigan guide titled *Stage Exit Process Guide* provides a procedure and sample report form that can be used for stage exits. This document is available on the MDIT SUITE website.

References: Chapter 2.0, Lifecycle Model, *Quality Reviews* provides an overview of the Quality Reviews to be conducted on a project.

Bibliography: The following materials were used in the preparation of the Functional Design Stage chapter.

1. Barker, Richard, *CASE*METHOD,* Tasks and Deliverables, 1990. pp. 5-10 and 5-11.

2. Gane, Chris and Sarson, Trish, *Structured Systems Analysis: Tools and Techniques*, Prentice-Hall, Inc., Englewood Cliffs, 1979.

3. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Guide to Software Design Descriptions*, IEEE Std 1016.1-1993, New York, 1993.

4. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Guide to Software Requirements Specifications*, ANSI/IEEE Std 830-1998, New York, 1998.

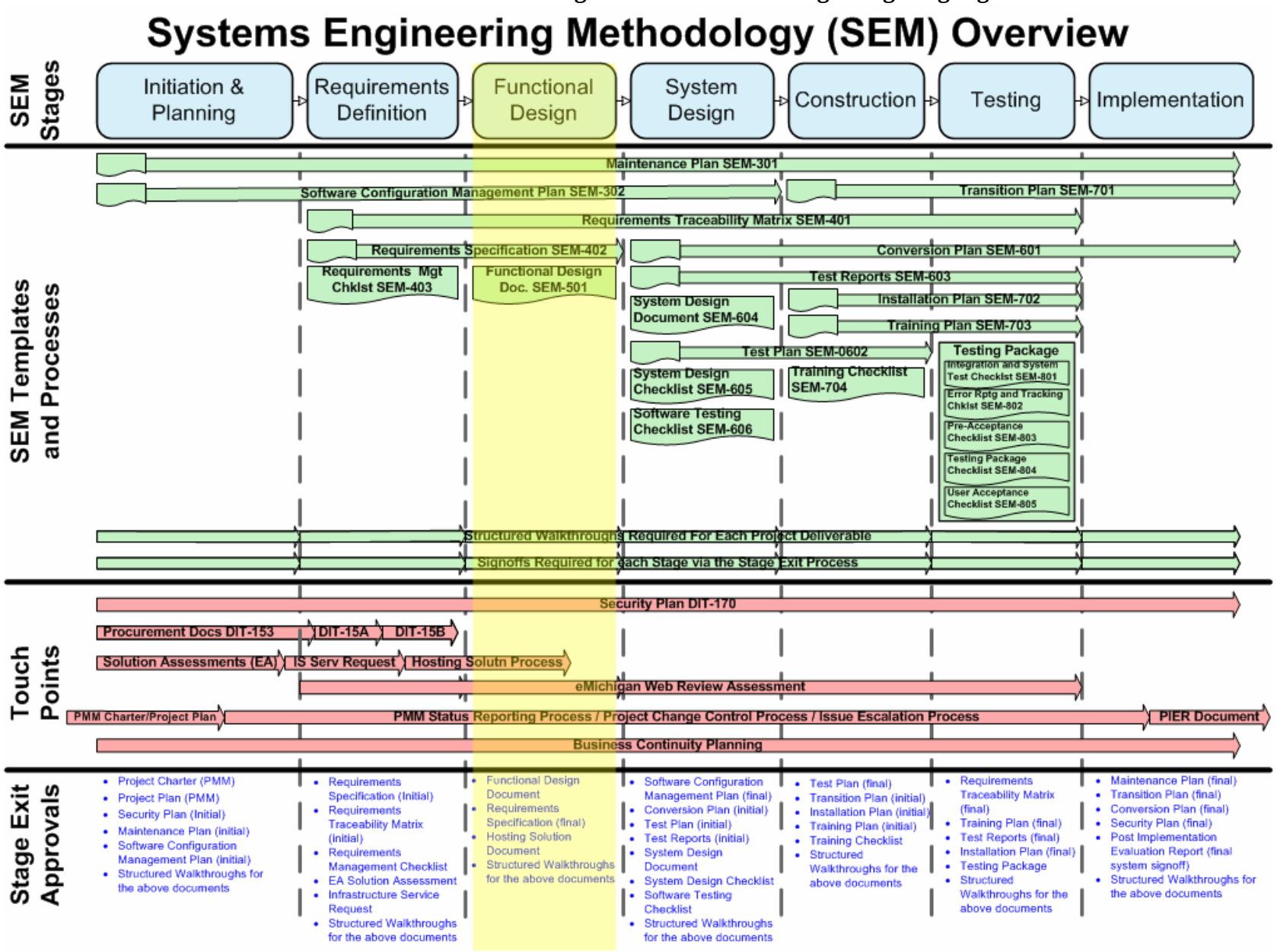
5. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Developing Software Lifecycle Processes*, IEEE Std 1074-1991, New York, 1992.

6. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Software Verification and Validation Plans*, ANSI/IEEE Std 1012-1998, New York, 1998.

7. U.S. Department of Defense, Military Standard, *Technical Reviews and Audits for Systems, Equipments, and Computer Software*, MIL-STD-1521 B, 1985. pp.5, 33-52.

8. U.S. Social Security Administration, Office of Systems, *Software Engineering Technology (SET) Manual*, Volume 1, 1990.

Exhibit 5.0-1 SEM Overview Diagram – Functional Design Stage Highlighted



Activity: **5.1 Determine System Structure**

Responsibility: Project Team Analysts

Description: A hierarchical approach is useful for determining the structure and components of the product. System decomposition is one hierarchical approach that divides the system into different levels of abstraction. Decomposition is an iterative process that continues until single purpose components (i.e., design entities or objects) can be identified. Decomposition is used to understand how the product will be structured, and the purpose and function of each entity or object.

The goal of the decomposition is to create a highly cohesive, loosely coupled, and readily adapted design. A design exhibits a high degree of cohesion if each design entity in the program unit is essential for that unit to achieve its purpose. A loosely coupled design is composed of program units that are independent or almost independent.

Several reliable methods exist for performing system decomposition. Select a method that enables the design of simple, independent entities. Functional and object-oriented design are two common approaches to decomposition. These approaches are not mutually exclusive. Each may be applicable at different times in the design process.

Tasks: The system decomposition activity includes the following tasks.

5.1.1 Identify Design Entities
5.1.2 Identify Design Dependencies

Task: **5.1.1 Identify Design Entities**

Description: Design entities result from a decomposition of the product requirements. A design entity is an element (or object) of a design that is structurally and functionally distinct from other elements and is separately named and referenced. The number and type of entities required to partition a design are dependent on a number of factors, such as the complexity of the product, the design method used, and the development environment. The objective of design entities is to divide the product into separate components that can be coded, implemented, changed, and tested with minimal effect on other entities.

Attributes: A design entity attribute is a characteristic or property of a design entity. It provides a statement of fact about an entity. The following are common attributes that should be considered for each design entity.

- Assign a unique name to each entity.
- Classify each entity into a specific type. The type may describe the nature of the entity, such as a subprogram or module; or a class of entities dealing with a particular type of information.
- Describe the purpose or rationale for each entity. Include the specific functional and performance requirements for which the entity was created.
- Describe the function to be performed by each entity. Include the transformation applied to inputs by the entity to produce the desired output.
- Identify all of the external resources that are needed by an entity to perform its function.
- Specify the processing rules each entity will follow to achieve its function. Include the algorithm used by the entity to perform a specific task and contingency actions in case expected processing events do not occur.
- Describe the data elements internal to each entity. Include information such as the method of representation, format, and the initial and acceptable values of internal data. This description may be provided in the data dictionary.

Work Product: Maintain a record of all design entities. The records will be integrated into the Functional Design Document.

Review Process: Schedule structured walkthroughs to verify that the design entities are correct, complete, and possess the required attributes.

Task: **5.1.2 Identify Design Dependencies**

Description: Design dependencies describe the relationships or interactions between design entities at the module, process, and data levels. These interactions may involve the initiation, order of execution, data sharing, creation, duplication, use, storage, or destruction of entities.

Identify the dependent entities of the system design, describe their coupling, and identify the resources required for the entities to perform their function. Also define the strategies for interactions among design entities and provide the information needed to perceive how, why, where, and at what level actions occur.

Dependency descriptions should provide an overall picture of how the product will work. Data flow diagrams, structure charts, and transaction diagrams are useful for showing the relationship among design entities.

The dependency descriptions may be useful in producing the system integration plan by identifying the entities that are needed by other entities and that must be developed first. Dependency descriptions can also be used to aid in the production of integration test cases.

Work Product: Add specific dependency information to the design entity records. The records will be integrated into the Functional Design Document.

Review Process: Schedule structured walkthroughs to verify that the design entities and dependencies are correct, complete, and possess the required attributes.

Activity: **5.2 Design Content of System Inputs and Outputs**

Responsibility: Project Team Analysts

Description: Design the content and format for each of the product inputs and outputs based on the system input and output requirements identified during the Requirements Definition Stage. Involve the system owner and users in the design process to make certain that their needs and expectations are being met.

Procedure: Use the following procedure to implement the design process.

- Identify the types of electronic and printed input that will be accepted by the product, such as data entered manually from source documents and files or records extracted from other systems.
- Identify the types of electronic and printed output that will be produced by the product; such as data, records, or files; screen displays; and printed reports. Also, identify the output that will be exported to other systems.
- Identify the specific input and output items that already exist and the items that will be created for input or output as part of the product.
- Assign a name to each type of input and output and describe each item from a functional perspective.
- Identify the owner/originator of each type of input and output.
- Identify the frequency of each type of input and output.
- Design the content and format for each new input and output item or modify the format of existing items that must be changed to accommodate the new product.

Work Product: Document the design for the system inputs and outputs in accordance with the project design standards. Discuss the designs with the system owner and users and submit completed designs for their review and approval. The approved designs will be incorporated into the Functional Design Document.

Review Process: Schedule a structured walkthrough to verify that the system input and output designs are correct and complete and describe the designs in a manner that can be understood by the system owner and users.

Activity: **5.3 Design User Interface**

Responsibility: Project Team Analysts

Description: Design a user interface that is appropriate for the users, content, and operating environment for the product. Determine interface levels for all categories of users. For interactive user environments, prototype the user interface. Arrange for users to experiment with the prototypes so that design weaknesses in the interface can be identified and resolved early. Use prototypes to gain user acceptance of the interface.

MDIT standards for e-government applications and relevant State of Michigan standards should be used to specify the user interface for every product developed.

Basic Principles: The following basic principles can help improve the product user interface when there is graphical, command-based, menu-driven, or block mode features.

- Give users control. Let them choose actions to perform.
- Give users feedback and progress reports. Tell them when the system is working and when an action is completed.
- Be consistent in the format and wording of text.
- Keep it simple. White space is as important on the screen as on the printed page. Reduce screen clutter.
- Put information where it can be easily seen; avoid information in corners or borders.
- Limit the amount of information users must know. Offer choices instead of making users remember and manually enter information. Provide defaults, and make sure they are logical and satisfy a large number of users.
- Offer shortcuts. Keyboard shortcuts (e.g., hot keys) and command abbreviations help experienced users work more quickly.
- Help users get out of trouble. Provide messages that are understandable and that offer solutions.
- Let users reverse their actions. If an action will destroy something, identify the object of destruction and wait for a response.

Tasks: The following tasks are involved in specifying the user interface.

5.3.1 Design Menu Hierarchy
5.3.2 Design Data Entry Screens
5.3.3 Design Display Screens
5.3.4 Design Online Help
5.3.5 Design System Messages

Task: **5.3.1 Design Menu Hierarchy**

Description: Use the following guidelines to improve the design of menu hierarchies.

- Choose an organizing principle for the menu options, such as:
    - Expected frequency of use
    - Logical sequence of operations
    - Alphabetical order (should be used for horizontal word menus with five or more words)
- Put a meaningful title at the top of every menu.
- For full-screen menus, provide symmetric balance by centering the title and the menu options around the center axis of the screen.
- To facilitate scanning, put blank lines between logical groupings of menu options and after about every fifth option in a long list.
- Limit the number of menu choices to one screen.
- Select icons that are intuitive to the function they represent.
- Use a menu option selection method that is consistent with the technology available at the user's workstation and the size of the product being designed, such as:
    - Numbers
    - Letters or letter combinations
    - Cursor movement
- Provide a way for the user to leave the menu without performing any action. Be sure that the option to leave the menu describes the consequences of its selection.
- Words used for menu options should follow these rules:
    - Use words that clearly and specifically describe what the user is selecting.
    - Use common English words rather than computer or technical jargon. When space permits, spell out words completely.
    - Use simple, active verbs to tell users what actions will result from their choice. Try to start each option with a verb.

  - Use parallel construction to describe the options.

- Minimize the highlighting used on a menu. Highlighting should be limited to situations where the user needs to know that there is an exception to the normal practice.

- Do not require the user to enter leading or trailing blanks or zeros, and do not include a default value on a menu.

- Display the menu options in mixed letters (i.e., upper and lower case).

- Organize menu hierarchies according to the tasks users will perform, rather than the structure of the modules.

Work Product: Document the design for the menu hierarchy in accordance with the project design standards. Discuss the design with the system owner and users and submit the completed design for their review and approval. The approved design will be incorporated into the Functional Design Document.

Review Process: Conduct a structured walkthrough to ensure that the menu hierarchy design is complete, logical and describes the design in a manner that can be understood by the system owner and users.

Task: **5.3.2 Design Data Entry Screens**

Description: Use the following guidelines to improve the design of data entry screens.

- When the user must transcribe data directly from a source document to the screen, the layout of the screen should be similar to the layout of the source document.
- Group data fields into logical categories on the screen; provide a header that describes the contents of each category.
- Make areas of the screen that are not needed for data entry or commands inaccessible to the user.
- Do not require the user to enter information that is already available to the software or can be computed by it.
- Do not require the user to enter dimensional units, leading or trailing blanks, or zeros.
- Allow the user to enter data by character replacement.
- Put a caption describing the data to be entered adjacent to each data field; incorporate memory joggers into the caption.
- Justify data entries automatically.
- Display default values in data fields when appropriate.
- Provide context-sensitive help for data entry fields.

Work Product: Document the designs for the data entry screens in accordance with the project design standards. Discuss the design with the system owner and users and submit the completed designs for their review and approval. The approved designs will be incorporated into the Functional Design Document.

Review Process: Conduct a structured walkthrough to assure that the data entry screen designs are consistent, complete, and logical and describe the designs in a manner that can be understood by the system owner and users.

Task: **5.3.3 Design Display Screens**

Description: Use the following guidelines to design display screens that are easy to use and understand.

- Put a title on every display screen. The title should clearly and specifically describe the contents of the screen.
- Display only information that the user needs to know.
- Display data to the user in directly usable form.
- Provide symmetric balance to displays by centering titles and headings and by placing information on both sides of the center axis.
- Every display should indicate how to exit from the screen. Use consistent exit procedures.
- When the display continues over multiple screens, the screen should indicate where the user is in the display (e.g., Screen 1 of 3).
- Data fields need to be grouped into logical categories or according to the structure of a source document (when there is one).
- Be consistent in the use of words and special characters.
- Display text conventionally in mixed letters (i.e., upper and lower case) and with appropriate punctuation. Avoid all uppercase letters. Put a blank line between paragraphs.
- Left justify text, and leave a ragged right margin.
- Avoid hyphenation of words between lines.
- Use abbreviations and acronyms only when they are significantly shorter than the full text and when they will be understood by the user.
- Be consistent with the format of information being displayed.
- Consider the skills of the users and the information they will manipulate when information is displayed in multiple windows.

Table and List Guidelines: Use the following guidelines to improve the design of online tables and lists.

- Put a meaningful label on the columns and, if appropriate, the rows of tables and lists. Continue the labels when a table or list extends over more than one screen.
- If data items are scrolled, the labels should be fixed on the screen and not be part of the scrolled area (they remain in place as the body of the table or list changes).
- If data items are continued on subsequent screens, the labels should be added to each screen.
- Arrange the items in a table or list in some recognizable order to facilitate scanning.
- Put items in a multiple column list in vertical columns that are read from left to right on the screen.
- Left justify columns of alphabetic data; right justify columns of numeric data or align them by the decimal point or other delimiter.
- Insert a blank line after about every fifth row in a long column.
- Insert a minimum of two spaces between the longest item in a column and the beginning of the next column.
- Start with a one (1) not a zero (0) when listed items are labeled by number.

Work Product: Document the design for the display screens in accordance with the project design standards. Discuss the designs with the system owner and users and submit the completed designs for their review and approval. The approved designs will be incorporated into the Functional Design Document.

Review Process: Conduct a structured walkthrough to ensure that the display screen designs are consistent, complete, logical complete, logical and describe the designs in a manner that can be understood by the system owner and users.

Task: **5.3.4 Design Online Help**

Description: Online help is typically requested by users when they want to perform a new, complex, or infrequently used procedure, or when they do not know what else to do. The text of online help messages needs to be planned, drafted, and evaluated as carefully as print documentation. In addition, the layout and format of online help must be designed to deal with the special constraints imposed by the video screen.

Use online help to explain concepts, procedures, messages, menu choices, commands, words, function keys, and formats. Work with the users to identify the level of detail needed for online help. Determine whether the users need a one-line message at the bottom of the screen or a full online explanation with successive levels of detail.

Effective online help messages tell users what the product is doing, where they are in the sequence of screens, what options they have selected, and what options are available.

Guidelines: The following guidelines can improve the design of online help.

- Write online help messages in plain English.
    - Straightforward and reads as if it were spoken.
    - Clear, direct, and simple.
    - Effectively organized with a concern for what users need to know.
- Address the user directly as "you"; use the active voice.
- Use simple action verbs to describe procedures. Do not use nouns to replace pronouns, verbs, and adjectives.
- Describe procedures in logical order.
- Avoid computer terms or other jargon, such as:
    - Terms that are unique to the computer profession or to a particular company.
    - Terms that have a common meaning outside of the data processing environment, but a special meaning within it, such as *boot*, *abort*, *default*, and *utility*.
    - Terms that are created to describe some special function, such as *ungroup* and *de-archive*.

- Avoid humor in online documentation.
- Write in short complete sentences and paragraphs and use proper punctuation.
- Write sentences in the positive or simple negative. Avoid the passive voice and do not use double negatives.
- Use bullets, numbered lists, and tables to make it easier to find the most important information. Leave ample open space.
    - Use bulleted lists to explain options. Whenever a sentence lists options with commas between them, consider breaking up the text into a bulleted list.
    - Use numbered lists to show the steps in a process.
    - Use a table to explain two or more categories of information.
- Use examples to show users what they should enter and what the results will look like.
- Do not expect users to read more than about three screens of help at one time.
- Provide an orientation to the structure of the product.
- Whenever possible display help text on the screen with the function or task that is being performed.
- Provide a direct route back to the function or task being performed.

Work Product: Document the design for online help in accordance with the project design standards. Discuss the design with the system owner and users and submit the completed design for their review and approval. The approved design will be incorporated into the Functional Design Document.

Review Process: Conduct a structured walkthrough to ensure that the online help design is consistent, complete, and logical and describe the design in a manner that can be understood by the system owner and users.

Task: **5.3.5 Design System Messages**

Description: System messages are the various types of information that the system provides to the user such as status messages, user prompts, and error messages.

Status Messages: Status messages are important for giving users the feeling they are in control of the system. They tell users what the system is doing, where they are in the sequence of screens, what options they have selected, and what options are available.

User Prompts: Prompts inform the user to type data or commands or to make a simple choice.

- Use prompts to ask the user to make a simple choice or to enter data or commands. Be as specific as possible.
- Include memory aids in the prompt to help users type a response in the proper format and order, initiate infrequently used processes, or clearly identify exceptions to normal practice.
- When defaults are allowed with prompts, indicate clearly which default value will be initiated.

Error Messages: Error messages should allow users to recover from mistakes by making it clear what the mistake was and how to correct it. Error messages need to be specific about why a mistake was made.

- Design the product to check for obvious errors.
- Be as specific as possible in describing the cause of an error. Do not use error codes.
- Do not assign blame to the user or the product in an error message. Use a neutral tone.
- Whenever possible, the error message should indicate what corrective action the user needs to take.
- Be consistent in the format, wording, and placement of messages.
- Consider describing error messages at more than one level of detail.

Work Product: Document the design for the system messages in accordance with the project design standards. Discuss the designs with the system owner and users and submit the completed designs for their review and approval. The approved designs will be incorporated into the Functional Design Document.

Review Process: Conduct a structured walkthrough to ensure that the system message designs are consistent, complete, and logical and describe the designs in a manner that can be understood by the system owner and users.

Activity: **5.4 Design System Interfaces**

Responsibility: Project Team Analysts

Description: Develop a design depicting how the product will interface with other systems based on the system interface requirements identified in the Requirements Definition Stage. Submit the applicable interface designs for review by the system owner or system administrator for each system that will interface with the product. Any incompatibilities with the interfaces will be identified early in the design process and corrective actions can be initiated to assure each interface is properly designed and coded.

Sample Issues: The following list provides some of the issues that should be considered when designing the system interfaces.

- System inputs and outputs
- Method of interface
- Volume and frequency of data
- Platform of interfacing system
- Format of data
- Automatic or manual initiation of interface
- Need for polling device(s)
- Verification of data exchange
- Validation of data

Work Product: Document the design(s) for the system interfaces in accordance with the project design standards. Discuss the designs with the system owner and users and submit completed designs for their review and approval. The approved designs will be incorporated into the Functional Design Document.

Review Process: Schedule a structured walkthrough to verify that the system interface designs are correct, complete, and logical and describe the design in a manner that can be understood by the system owner and users.

Activity: **5.5 Design System Security Controls**

Responsibility: Project Team Analysts and MDIT Office of Enterprise Security liaison

Description: Design the security controls that will be incorporated into the product based on the security and access requirements identified during the Requirements Definition Stage. Design the security controls in conjunction with the site or system owner and the MDIT Office of Enterprise Security liaison.

Procedure: Use the following procedure to implement the design process.

- Identify the users and organizations that will have access to the product. Indicate what access restrictions they will have. All persons in a work area may not have the same security access level. Measures should be taken to assure that sensitive materials and systems requiring protection are not accessed by unauthorized individuals.

- Identify controls for the product, such as the user identification code for system access and the network access code for the network on which the product will reside.

- Identify whether access restrictions will be applied at the system, subsystem, transaction, record, or data element levels. Sensitive information must be protected in accordance with State of Michigan directives.

- Identify physical safeguards required to protect hardware, software, or information from natural hazards and malicious acts.

- Identify communications security requirements.

Work Product: Document the design for the system security controls in accordance with the project design standards. Discuss the design with the system owner and users and submit the completed design for their review and approval. The approved design will be incorporated into the Functional Design Document.

Review Process: Schedule a structured walkthrough to verify that the system security controls are correct, complete and describe the controls in a manner that can be understood by the system owner and users. Include the MDIT Office of Enterprise Security liaison in the walkthrough.

Activity: **5.6 Build Logical Model**

Responsibility: Project Team Analysts

Description: The logical model defines the flow of data through the system and determines a logically consistent structure for the system. Each module that defines a function is identified, interfaces between modules are established, and design constraints and limitations are described. The focus of the logical model is on the real-world problem or need to be solved by the product.

A logical model has the following characteristics:

- Describes the final sources and destinations of data and control flows crossing the system boundary rather than intermediate handlers of the flows.
- Describes the net transfer of data across the system boundary rather than the details of the data transfer.
- Provides for data stores only when required by an externally imposed time delay.

When building a logical model, the organization of the model should follow the natural organization of the product's subject matter. The names given to the components of the model should be specific. The connections among the components of the model should be as simple as possible.

Work Product: The logical model should be documented in user terminology and contain sufficient detail to obtain the system owner's and users' understanding and approval. Use data flow diagrams to show the levels of detail necessary to reach a clear, complete picture of the product processes, data flow, and data stores.

Maintain the logical model and data flow diagrams for incorporation into the Functional Design Document. Keep the logical model and diagrams up-to-date. They will serve as a resource for planning enhancements during maintenance, particularly for enhancements involving new functions.

Review Process: Schedule a structured walkthrough to verify that the logical model is correct, logical, and complete.

Activity: **5.7 Build Data Model**

Responsibility: Project Team Analysts

Description: A data model is a representation of a collection of data objects and the relationships among these objects. The data model is used to provide the following functions:

- Transform the business entities into data entities.
- Transform the business rules into data relationships.
- Resolve the many-to many relationships as intersecting data entities.
- Determine a unique identifier (key) for each data entity.
- Add the attributes (facts) for each data entity.
- Document the integrity rules required in the model.
- Determine the data accesses (navigation) of the model.

Work Product: The data dictionary is developed in this stage. Its purpose it to catalog every known data element used in the user's work and every system-generated data element. Data elements are documented in detail to include attributes, known constraints, input sources, output destinations, and known formats.

The data dictionary can serve as a central repository of information for both developers and end users. The dictionary can include business rules, processing statistics, and cross-referencing information for multiple vendor environments.

To expand the data dictionary, define, analyze, and complete data definitions using the following steps.

- Identify data needs associated with various system features.
- Match (verify) data needs with the data dictionary.
- Match the data dictionary with specific data structures.
- Create data record layouts.
- Ensure that all data can be maintained through add, change, or delete functions.

The data dictionary may be further refined in the System Design Stage to complete the information on data elements, entities, files, physical characteristics, and data conversion requirements.

Sample Attributes: The following is a sample of the type of attributes (information) that should be included for each element in a data dictionary.

Long data name (full name)
Short data name (abbreviation)
Alias
Data definition
Owner(s)
Occurrence(s)/key
Program mode
Input source(s) (e.g., screens, external interfaces, system generated)
Output destination(s) (e.g., screens, reports, external interfaces)
Values/meanings
Protection/security
Default value
Length/precision
Character set (type)
Format
Range
Surface edits
Remarks

Review Process: Schedule a structured walkthrough to verify that the data dictionary is correct and complete. The data model for a software application should be validated against any MDIT or site specific data model.

Activity: **5.8 Develop Functional Design**

Responsibility: Project Team

Description: The functional design describes how the product will be structured to satisfy the requirements identified in the Requirements Specification. It is a description of the structure, components, interfaces, and data necessary before coding can begin.

The functional design is a model or representation of the product that is used primarily for communicating design information to facilitate analysis, planning, and coding decisions. It represents a partitioning of the system into design entities and describes the important properties and relationships among those entities. Design descriptions may be produced as documents, graphic representations, formal design languages, records in a database management system, and Computer Aided Systems Engineering (CASE) tool dictionaries.

Within the functional design, the design entities can be organized and presented in any number of ways. The goal of this activity is to compile the design entities and their associated attributes in a manner that facilitates the access of design information from various viewpoints (e.g., project management, software configuration management, quality assurance, and testing). Also, the design entities and their attributes must be described in terms that are understandable to the system owner and users.

Work Product: Major work products are the Functional Design and the revised Requirements Traceability Matrix. Each requirement identified in the Requirements Specification must be traceable to one or more design entities. This traceability ensures that the product will satisfy all of the requirements and will not include inappropriate or extraneous functionality. Expand the Requirements Traceability Matrix developed in the Requirements Definition Stage to relate the functional design to the requirements.

Review Process: Conduct a structured walkthrough of the Functional Design and the Requirements Traceability Matrix.

Tasks: The following tasks are involved in developing the functional design.

5.8.1 Develop Functional Design Document
5.8.2 Conduct Functional Design Review

Task: **5.8.1 Develop Functional Design Document**

Description: The Functional Design Document defines the functions of the system in user terminology and provides a firm foundation for the development of the system design. The Functional Design Document should be written from the system owner/users' perspective. This document provides the owner/users with an opportunity to review and provide input to the product design before system design work is completed.

Work Product: Prepare a draft Functional Design Document. Use the designs developed for inputs, outputs, user and system interfaces, and security controls as input to this document. Submit the draft document to the system owner and users for their review and approval. After making the changes needed to resolve problems found during the review, the approved Functional Design Document becomes an official agreement and authorization to use the functional design as the basis for developing the system design. Place a copy of the approved Functional Design Document in the Project File.

Review Process: Conduct structured walkthroughs as needed to assure that the Functional Design Document is accurate, complete, and describes the functional design in a manner that can be understood by the system owner and users.

Resource: A template of the Functional Design Document is available on the MDIT SUITE website.

Task: **5.8.2 Conduct Functional Design Review**

Description: The Functional Design Review is a formal technical review of the basic design approach. The primary goal of the Functional Design Review is to demonstrate the ability of the information system design to satisfy the project requirements. The review should be a series of presentations by the project team to the system owner, users, functional area points-of-contact, and quality assurance representative. Vendors may be invited to participate in the Functional Design Review when an off-the-shelf software product or hardware item is being considered for the system architecture.

Conduct the Functional Design Review to perform the following verifications.

- Evaluate the progress, technical adequacy, and risk mitigation of the selected design approach. Determine whether the approved design approach is being followed by the project team.

- Evaluate the progress, technical adequacy, and risk mitigation of the selected test approach. Review the following items:

    - System Test Requirements from the Requirements Specification document
    - Organization and responsibilities of group conducting tests
    - Planned format, content, and distribution of test reports
    - Planned resolution of problems and errors identified during testing
    - Retest procedures
    - Change control and configuration management of test items
    - Special test tools not required as deliverables

- Evaluate the techniques to be used to meet quality assurance requirements.

- Establish the existence and compatibility of the physical and functional interfaces.

- Determine whether the functional design embodies all of the product requirements.

- Verify that the design represents a system that can meet the functional, data, and interface requirements.

- Review the planned user interfaces to the system. Examples of the types of design information to review:

    - Operating modes for each display station. For each mode, the functions performed, and the displays and controls used.

- The format and content standards for each screen (e.g., data locations, spaces, abbreviations, the number of digits, all special symbols, alert mechanisms).
- Control and data entry devices and formats (e.g., keyboards, special function keys, and cursor control).
- The format of all data inputs and provisions for error detection and correction.
- The format for all status and error messages and data printouts (e.g., formats, headings, data units, abbreviations, spacing, and columns).

- Demonstrate any rapid design prototypes used to make design decisions.
- Identify potential high risk areas in the design and any requirements changes that could reduce risk.
- Review to assure that consideration has been given to optimizing the maintainability and maintenance aspects of the product.

Review Items: The following items should be considered for review and evaluation during the Functional Design Review. Be prepared to discuss in technical detail any of these items within the scope of the review.

- Functional flows. Indicate how the system functional flows map the software and interface requirements to the individual high-level components of the product.
- Storage allocation data. Describe the manner in which available storage is allocated to individual components. Timing, sequencing requirements, and relevant equipment constraints used in determining the allocation should be included.
- Control functions. Describe the executive control and start/recovery features of the product.
- Component structure. Describe the high-level structure of the product, the reasons for choosing the components, the development technique that will be used within the constraints of available computer resources, and any support programs that will be required in order to develop and maintain the product and allocated data storage.

- Security. Identify the security requirements and provide a description of the techniques to be used for implementing and maintaining security within the product.

- Information systems engineering facilities. Describe the availability, adequacy, and planned utilization of the information systems engineering facilities including both Government-provided and commercially available facilities.

- Information systems engineering facility versus the operational system. Describe any unique design features that exist in the functional design in order to allow use within the information systems engineering facility that will not exist in the operational product. Provide information on the design of support programs not explicitly required for the operational system that will be generated to assist in the development of the product.

- Development tools. Describe any special tools (e.g., simulation, data reduction, or utility tools) that are not deliverables, but are planned for use during systems development.

- Test tools. Describe any special test systems, test data, data reduction tools, test computer software, or calibration and diagnostic software that are not deliverables, but are planned for use during development.

- Commercial resources. Describe commercially available computer resources, including any optional capabilities (e.g., special features, interface units, special instructions, controls, formats). Identify any limitations of commercially available equipment (e.g., failure to meet user interface, safety, and maintainability requirements) and identify any deficiencies.

- Existing documentation. Maintain a file and have available for review any existing documentation supporting the use of commercially available computer resources.

- Support resources. Describe the resources necessary to support the product during engineering, installation, and operational state (e.g., operational and support hardware and software personnel, special skills, human factors, configuration management, testing support, documentation, and facilities/space management).

- Standards. Describe any standards or guidelines that must be followed.

- Operation and support documentation. Describe the documentation that will be produced to support the operation and maintenance of the product.

Work Product: The work product is the Functional Design Document. The review of this document will result in one of the following outcomes:

- Approval - indicates that the functional design is satisfactorily completed.
- Contingent Approval - indicates that the functional design is not considered accomplished until the satisfactory completion of identified action items.
- Disapproval - indicates that the functional design is inadequate. Another Functional Design Review is required, once specified changes to the functional design are completed.

After making changes needed to resolve problems found during the review, the now Approved Functional Design Document becomes an official agreement and authorization to use the functional design as the basis for developing the system design.

Activity: **5.9 Select System Architecture**

Responsibility: Project Team

Description: Due to the complexities inherent with contracting and procurement processes and establishing hosting environments, the need exists to select the system architecture at this stage of the SEM.

When the system architecture for the product has not been predetermined by the existing IT environment of the system owner and users, evaluate system architecture alternatives to determine which one has the best, cost-effective solution that satisfies the project requirements. By conducting the MDIT Enterprise Architecture (EA) Solution Assessment process, the final system architecture will be determined.

"Cost effective solution" does not imply the least expensive alternative. The best, cost effective solution is the alternative that does the best job of satisfying the project requirements, assures the highest quality product, and provides for an adequate return on investment in a timeframe that is acceptable to the system owner.

Select the specific hardware, software, database management system, and communication facilities based on the following types of considerations.

- MDIT Enterprise Architecture or site-specific information architecture guidelines or standards
- Hardware and software that emphasizes simplicity, flexibility, ease of operation and maintenance
- Cost to procure and maintain potential environment
- Backup and recovery procedures
- Selection of a distributed or centralized processing environment
- Communication requirements
- Data configuration

Obtain support from functional area points-of-contact to aid in the architecture evaluation process. Consultations and input may be helpful from system and database administrators, local area network administrators, operations personnel, system developers, and communication experts.

Tasks: The following tasks are involved in selecting a system architecture.

5.9.1 Evaluate System Architecture Alternatives
5.9.2 Recommend System Architecture

Task: **5.9.1 Evaluate System Architecture Alternatives**

Description: Consider system architecture alternatives within the site's enterprise architecture guidelines that enable the project objectives and requirements to be achieved. The selection of a system architecture depends on many factors such as the experience of the project team with each alternative and the availability of reusable components to facilitate the implementation of an alternative. MDIT Enterprise Architecture (EA) should be consulted during the selection process.

When investigating alternatives with EA, consider the following issues.

- Those functions or portions of functions that are to be automated and the functions that will be manual. Conduct an examination of *what* the automated portion of the project will encompass.
- The technical solution for the objectives. The determinations of *how* the product is to be designed; (e.g., online vs. batch, client-server vs. mainframe, Oracle vs. SQL Server).
- The system owner's and users' IT environment and the needs created by the technical solution. Consider any hardware and software that must be acquired, including system access software, operating system software, database management system, and communications facilities.

The following procedure provides one approach for evaluating the architecture alternatives.

- Conduct a Business Case Analysis to determine the most cost-effective alternative. On the benefits side, include the improvements over the current process being used to support the business application. On the costs side, include any degradation from current capabilities along with the rationale for allowing the degradation.
- Create and evaluate a data flow diagram for each alternative.
- Identify how users would interact with the features associated with each alternative (such as the generation of queries and reports).
- Create a list of the risks associated with each alternative and develop a plan for mitigating each risk.
- Compare the performance capabilities of each alternative. How fast will each alternative be able to process the user's work given a particular hardware resource. Performance is usually expressed in terms of throughput, run time, or response time.

Five factors that frequently affect performance include:

- Number of intermediate files in a system (parked data between programs)
- Number of times a given file is passed
- Number of seeks against a disk file
- Time spent in calling programs and other system overhead
- Time taken to execute actual program

- Compare the security and access control features of each alternative. To what extent does the alternative provide security against human errors, machine malfunction, or deliberate mischief. Some common controls include:

  - Check digits on predetermined numbers
  - Batch control totals
  - Creation of journals and audit trails
  - Limited access to files

- Compare the ease with which each alternative allows the system to be modified to meet changing requirements, such as:

  - Fixing errors
  - Changing user needs
  - Mandatory/statutory modifications
  - Enhancements

Work Product: Maintain records on each alternative that is evaluated. Use this information to develop a summary of the system architecture alternatives. The summary will be integrated into the materials presented to the system owner when a system architecture recommendation is made. Place a copy of the records for each alternative and the summary in the Project File.

If a Business Case Analysis is conducted, prepare a report that describes the process used for the analysis, a summary of the alternatives considered and the results obtained, and place a copy in the Project File. The report will be integrated into the materials presented to the system owner when a system architecture recommendation is made.

Review Process: Conduct structured walkthroughs on records of each alternative that is evaluated.

Task: **5.9.2 Recommend System Architecture**

Description: Based on the results of the architecture alternatives evaluation, develop a recommendation for a system architecture that is cost-effective and will facilitate the achievement of the project requirements. This recommendation is contained in the MDIT Hosting Solution document. Prepare a presentation for the system owner and users that provides the following types of information to support the recommendation. MDIT EA and Technical and Data Center Services should be consulted during the selection process.

- Review the limitations or problems with any current manual or IT system that will be resolved by the product.
- Present the logical model for the product. Highlight new functions that would be incorporated.
- For each architecture alternative that was evaluated, present the following information.
  - A description of the alternative.
  - An overall data flow diagram showing how the alternative would be implemented.
  - The way the system would look to the users in terms of hardware, user interface, reports, and query facilities.
  - The estimated benefits of the alternative.
  - The estimated cost and time to implement the alternative.
  - A statement of the element of risk associated with the alternative.
- Present the recommended alternative and explain why it was selected.

Formal acceptance of the project team's recommendation by the system owner is required before the project can move forward. Any delay in making this decision could result in a slippage of the project schedule.

Work Product: Document the project team's recommendation for the most cost-effective and viable architecture alternative. Provide a summary of each alternative that was evaluated. Describe the rationale for proposing the recommended architecture. Describe the impact of this alternative on the system owner and users organization(s) and other systems. Include any background information that was relevant to the decision process. Provide access and/or copies of DIT-015a/DIT-015b and other related templates to appropriate users/DIT staff for preparation.

Review Process: Conduct a structured walkthrough to assure that the most cost-effective and viable architecture alternative is being recommended.

Approval: Present the project team's recommendation for the system architecture to the system owner and users. The recommendation should be delivered as both a document and as a presentation. Place a copy of the MDIT Hosting Solution Document in the Project File.

Chapter: **6.0 System Design Stage**

Description: The goal of this stage is to translate the user-oriented functional design specifications into a set of technical, computer-oriented system design specifications; and to design the data structure and processes to the level of detail necessary to plan and execute the Construction and Implementation Stages. General module specifications should be produced to define what each module is to do, but not how the module is to be coded. Effort focuses on specifying individual routines and data structures while holding constant the structure and interfaces developed in the previous stage. Each module and data structure is considered individually during detailed design with emphasis placed on the description of internal and procedural details. The primary work product of this stage is a system design that provides a blueprint for the coding of individual modules and programs.

Input: The following items provide input to this stage:

SEM Templates:

- Functional Design document
- Maintenance Plan
- Requirements Specification
- Requirements Traceability Matrix
- Software Configuration Management Plan

PMM Templates:

- Project Plan
- Quality Management Plan
- Security Plan

Other Inputs:

- Data Dictionary

High-Level Activities: The remainder of this chapter is divided into sections that describe the specific high-level activities performed during this stage. These activities represent the minimum requirements for a large information systems engineering effort. Notes are provided, as applicable, to assist in customizing these lifecycle stage requirements to accommodate different sizes of information systems engineering efforts. The high-level activities are presented in the sections listed below.

6.5 Develop Conversion Plan
6.6 Develop System Design
6.7 Develop Program Specifications

Touch Points: The following touch points are involved in the System Design Stage:

- Contracts and Procurement
  - Contract Liaison involvement if contract issues arise
- E-Michigan
  - Continue to work with E-Michigan's webmaster, as appropriate, to ensure ADA compliance and Michigan.gov look and feel standards.
- Enterprise Architecture (EA)
  - Request exceptions as required
  - Complete EA Solution Assessment for the chosen solution and submit to EA for review/approval
- Infrastructure Services
  - Technical and Data Center Services Solutions Engineer involvement in completion of Hosting Solution document
  - Infrastructure Specialist involvement in establishing the construction environment
- Security
  - Review MDIT and Agency Security Policies
  - Review State and Federal Laws and Regulations
  - Revise Network Diagram and Data Flow Diagrams
  - Review Final Risk Analysis with OES recommended security controls

Output: Several work products are produced during this stage. The work products listed below are the minimum requirements for a large project. Deviations in the content and delivery of these work products are determined by the size and complexity of the project. Explanations of the work products are provided under the applicable activities described in the remainder of this chapter.

SEM Templates:

- Conversion Plan (*initial*)
- Maintenance Plan (*revised*)
- Requirements Traceability Matrix *(revised)*
- Software Configuration Management Plan (*final*)
- Software Testing Checklist (*final*)

- System Design Checklist (*final*)
- System Design Document (*final*)
- Test Plan *(initial)*
- Test Reports (*initial*)

PMM Templates:

- Project Plan *(revised)*
- Security Plan (*revised*)

A diagram showing the work products associated with each high-level activity is provided in *Exhibit 6.0-1, SEM Overview Diagram – System Design Stage Highlighted.*

Review the Project Plan for accuracy and completeness of all System Design Stage activities and make any changes needed to update the information.

Review Process: Quality reviews are necessary during this stage to validate the product and associated work products. The activities that are appropriate for quality reviews are identified in this chapter and Chapter 2.0, Lifecycle Model. In addition, a Critical Design Review is conducted once the System Design Document is developed. The time and resources needed to conduct the walkthroughs and Critical Design Review should be indicated in the project resources, schedule, and work breakdown structure.

Structured Walkthrough (SWT)
Requirements for a peer review or a more formal structured walkthrough are documented under *Review Process* at the end of each Task, Subtask, or Activity section in this stage. The State of Michigan guide titled *Structured Walkthrough Process Guide* provides a procedure and sample forms that can be used for SWTs. This document is available on the MDIT SUITE website.

Stage Exit
Schedule a Stage Exit as the last activity of the System Design Stage to enable the project approvers to review project deliverables and provide a concur/non-concur position to the project manager. The State of Michigan guide titled *Stage Exit Process Guide* provides a procedure and sample report form that can be used for stage exits. This document is available on the MDIT SUITE website.

References: Chapter 2.0, Lifecycle Model, *Quality Reviews* provides an overview of the Quality Reviews to be conducted on a project.

Bibliography: The following materials were used in the preparation of the System Design Stage chapter.

1. Barker, Richard, *CASE*METHOD,* Tasks and Deliverables, 1990. pp. 5-10 and 5-11.

2. Gane, Chris and Sarson, Trish, *Structured Systems Analysis: Tools and Techniques*, Prentice-Hall, Inc., Englewood Cliffs, 1979.

3. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Guide to Software Design Descriptions,* IEEE Std 1016.1-1993, New York, 1993.

4. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Guide to Software Requirements Specifications,* ANSI/IEEE Std 830-1998, New York, 1998.

5. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Recommended Practice for Software Design Descriptions,* IEEE Std 1016-1998, New York, 1998.

6. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Developing Software Lifecycle Processes,* IEEE Std 1074-1991, New York, 1992.

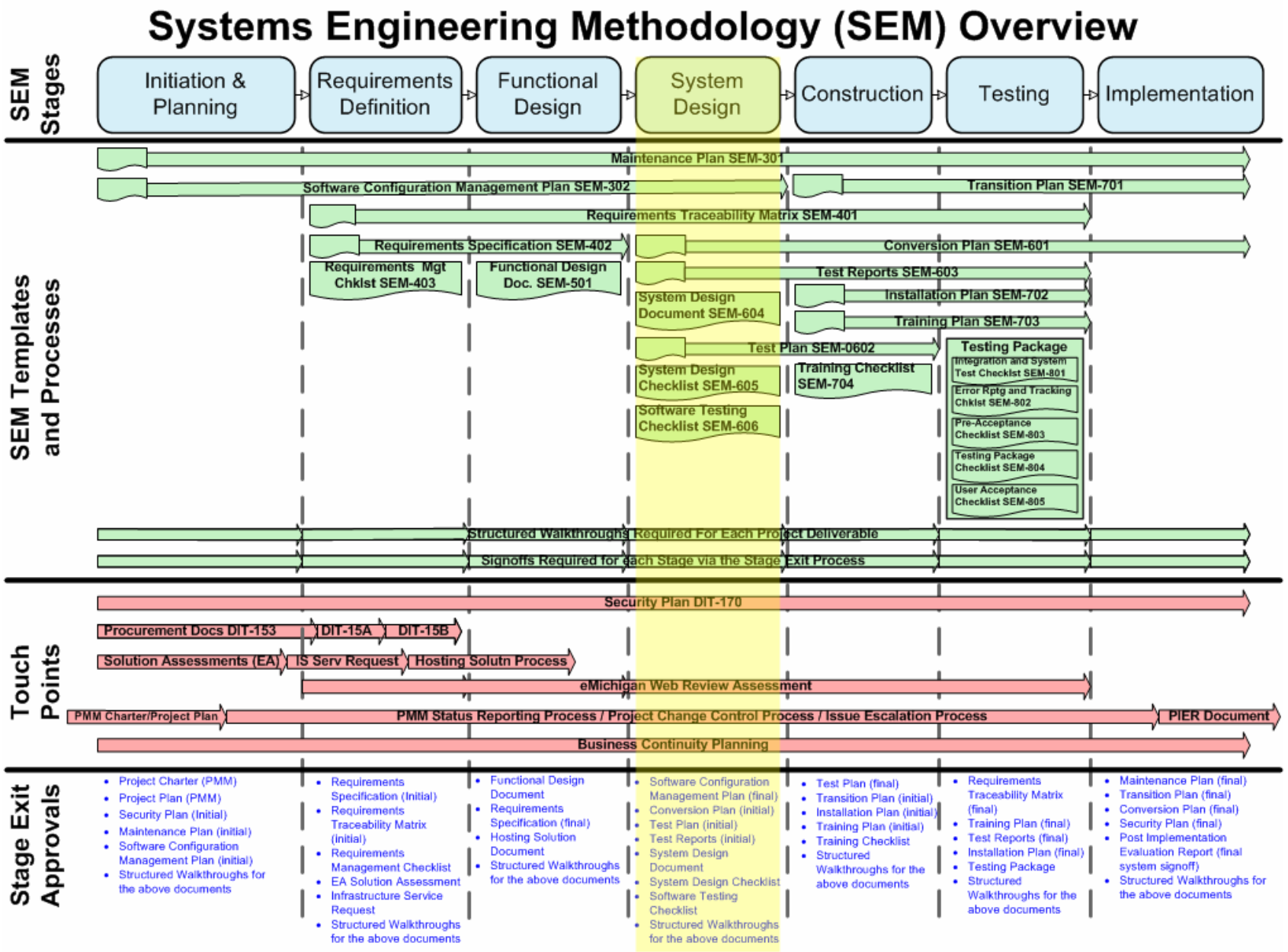
7. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Software Verification and Validation Plans,* ANSI/IEEE Std 1012-1998, New York, 1998.

9. U.S. Department of Defense, Military Standard, *Specification Practices,* MIL-STD-490 B, 1986. pp. 47-50.

10. U.S. Department of Defense, Military Standard, *Technical Reviews and Audits for Systems, Equipments, and Computer Software,* MIL-STD-1521 B, 1985. pp.5, 33-52.

11. U.S. Social Security Administration, Office of Systems, *Software Engineering Technology (SET) Manual,* Volume 1, 1990. Part 40 Design Stage.

Exhibit 6.0-1 SEM Overview Diagram – System Design Stage Highlighted



Activity: **6.1 Design Specifications for Modules**

Responsibility: Project Team

Description: During the Functional Design Stage, a decomposition of the product requirements resulted in a collection of design entities (or objects). In the System Design Stage, these design entities are grouped into the routines, modules, and programs that need to be developed or acquired as off-the-shelf or reusable software.

Expand the functional design to account for each major action that must be performed and each data object to be managed. Detail the design to a level such that each program represents a function that a developer will be able to code.

Procedure: Use the following procedure to design the module specifications.

- Identify a program for each action needed to meet each function or data requirement in the Requirements Specification and the data dictionary.
- Identify any routines and programs that may be available as reusable code or objects from existing applications or off-the-shelf software.
- Identify programs that must be designed and developed (custom-built). Assign a name to each program and object that is functionally meaningful. Identify the system features that will be supported by each program.
- Specify each program interface. Update the data dictionary to reflect all program and object interfaces changed while evolving from the functional to the system design.
- Define and design significant attributes of the programs to be custom-built.
- Expand the program interfaces to include control items needed for design validity (e.g., error and status indicators).
- Combine similar programs and objects. Group the design entities into modules based on closely knit functional relationships. Formulate identification labels for these modules.
- Show dependencies between programs and physical data structures (e.g., files and global tables). Avoid defining a program that not only needs data residing in a file or global table, but also depends on the physical structure or location of data.

- Change the design to eliminate features that reduce maintainability or reusability (i.e., minimize coupling between programs and maximize the cohesion of programs).

Work Product: Document the system design primarily in the form of diagrams. Supplement each diagram with text that summarizes the function (or data) and highlights important performance and design issues.

When using structured design methods, the design diagrams should:

- Depict the product as a top-down set of diagrams showing the control hierarchy of all programs to be implemented.
- Define the function of each program.
- Identify data and control interfaces between programs.
- Specify files, records, and global data accessed by each program.

When using object-oriented or data-centered design methods, the design diagrams should:

- Show the data objects to be managed by the product.
- Specify the program functions to be included within each object.
- Identify functional interfaces between objects.
- Specify files and records comprising each object.
- Identify relationships between data files.

Review Process: Conduct structured walkthroughs to assure that the custom-built routines and programs are correctly designed.

Activity: **6.2 Design Physical Model and Database Structure**

Responsibility: Project Team

Description: The physical model is a description of the dynamics, data transformation, and data storage requirements of the product. The physical model maps the logical model created during the Functional Design Stage to a specific technical reality. Care must be taken to retain in the physical implementation all of the capabilities inherent in the logical model.

The physical model frequently differs from the logical model in the following areas.

- Constraints imposed by the database management system - The logical data model may have different implementations in the selected database management system.
- Performance - Data redundancies, indices, and data structure changes may have to be introduced into the physical model to improve performance.
- Distributed processing - Possible network and multiple production hardware configurations may cause changes to the physical data model.

Designing the database structure converts the data requirements into a description of the master and transient files needed to implement the requirements. If the product will include a database, design the database in conjunction with the following database management features.

- Report writer and file processing capabilities
- Online query processing to retrieve data
- Automated data dictionary systems

Work Product: Document the physical model for incorporation into the System Design Document. Review the contents of the data dictionary entries and update to complete information on data elements, entities, files, physical characteristics, and data conversion requirements.

Review Process: Schedule structured walkthroughs to verify that the physical model and data dictionary are correct and complete.

Activity: **6.3 Develop Integration Test Considerations**

Responsibility: Project Team Developers

Description: The purpose of integration testing is to verify the integrity of a module (a cohesive set of programs) and its interfaces with other modules within the product structure. An integration test plan is developed to incorporate successfully unit-tested modules into the overall structure and to test each level of integration to isolate errors introduced by newly incorporated modules.

The number of integration levels, the classes of tests to be performed, and the order in which routines and builds are incorporated into the overall structure are addressed in the Integration Test portion of the overall Test Plan package. The following factors should be considered.

- Are routines to be integrated in a pure top-down manner or should builds be developed to test sub-functions first?
- In what order should major functions be incorporated?
- Is the scheduling of module coding and testing consistent with the order of integration?
- Is special hardware required to test certain routines?

Integration testing should include tests that validate the following functions.

- Verify each interface between the module and all other modules.
- Access each input message or command processed by the module.
- Check each external file or data record referenced by coding statements in the module.
- Output each message, display, or record generated by the module.

An important consideration during integration test planning is the amount of test software (e.g., drivers, test case generation) that must be developed to adequately test the required functionality. For example, it may be cost-effective to delay testing of a communication function until hardware is available rather than generate test software to simulate communication links.

Similarly, it may be better to include certain completed modules in the structure in order to avoid having to develop software drivers. These decisions are made on the basis of cost and risks.

Work Product: Develop draft Integration Testing procedures that address the following activities.

- Define the integration tests at each element level, stating objectives, what is to be tested, and verified. Testing is from the point of view of structure and function.
- Define all aspects of the formal interfaces that must undergo formal integration testing. Review interface requirements to ensure completeness, consistency, and effectiveness.
- Plan for test tools and software that must be developed to adequately test the required functionality.

Note: The Integration Test Considerations are incorporated in the Test Plan.

Review Process: Conduct a peer review or structured walkthrough to assure that the draft Integration Test section of the Test Plan is accurate and complete. The Integration Test section will be reviewed and revised as needed during the Construction Stage.

Activity: **6.4 Develop System Test Considerations**

Responsibility: Project Test Team

Description: The objectives of the system test process are to assure that the product adequately satisfies the project requirements; functions in the computer operating environment; successfully interfaces between user procedures, operating procedures, and other systems; and protects the software and data from security risks. The system should be tested under the same kind of daily conditions that will be encountered during regular operations. System timing, memory, performance, and security functions are tested to verify that they perform as specified. The functional accuracy of logic and numerical calculations are tested for verification under normal and load conditions.

Test data should be varied and extensive enough to enable the verification of the operational requirements. Expected output results should be included in the test plan in the form of calculated results, screen formats, hardcopy output, predetermined procedural results, warnings, error messages and recovery.

Detailed planning for the system testing helps to ensure that system acceptance will be successfully completed on schedule. The Software Testing Checklist should be completed during this process. When applicable, system testing must include the following types of tests.

- Performance tests that measure throughput, accuracy, responsiveness, and utilization under normal conditions and at the specified maximum workload.
- Stress tests to determine the loads that result in appropriate, non-recoverable, or awkward system behavior.
- Interface tests to verify that the system generates external outputs and responds to external inputs as prescribed by approved interface control documentation.
- Infrastructure and system recovery and reconfiguration tests.
- Verification that the infrastructure and system can be properly used and operated in accord with user's guides and operating instructions.
- Verification that the infrastructure and system meet their requirements for reliability, maintainability, and availability, including fault tolerance and error recovery.

- Verification of the effectiveness of error detection and analysis, and automated diagnostic tools.

- Demonstration that the infrastructure and system comply with their serviceability requirements such as accessibility, logistics, upgrades, diagnostics, and repair capabilities.

Work Product: Complete the Software Testing Checklist to ensure that all testing considerations have been addressed.

Develop draft System Test Considerations that describe the testing effort, provide the testing schedule, and define the complete range of test cases that will be used to assure the reliability of the product. The test cases must be complete and the expected output known before testing is started. The test considerations should address the following.

- Provide a definition of, and the objectives for, each test case.

- Define the test scenario(s) including the step-by-step procedure, the number of processing cycles to be tested or simulated, and the method and responsibility for feeding test data to the system.

- Define the test environment including the infrastructure and software environment under which the testing will be conducted.

- Identify and describe manual procedures, automated procedures, and test sites (real or simulated).

- Identify test tools and special test support needs (e.g., hardware and software to simulate operational conditions or test data that are recordings of live data).

- Identify responsibilities for conducting tests; for reviewing, reporting, and approving the results; and for correcting error conditions.

- Develop a requirements verification matrix mapping individual tests to specific requirements and specifying how each system requirement will be validated.

- Schedule for integrating and testing all components including adequate time for retesting.

Note: The System Test Considerations are incorporated into the Test Plan.

Review Process: Conduct peer reviews or structured walkthroughs to assure that each system test procedure is accurate, complete, and accomplishes the stated objectives. The System Test Considerations will be reviewed and revised as needed during the Construction Stage.

Activity: **6.5 Develop Conversion Plan**

Responsibility: Project Team

Description: If the product will replace an existing IT system, develop a Conversion Plan. The major elements of the Conversion Plan are to develop conversion procedures, outline the installation of new and converted files/databases, coordinate the development of file-conversion, and plan the implementation of the conversion procedures.

File conversion should include a confirmation of file integrity. Determine what the output in the new system should be compared with the current system, and ensure that the files are synchronized. The objective of file conversion is new files that are complete, accurate and ready to use.

Many factors influence data conversion, such as the design of the current and new systems and the processes for data input, storage, and output. Understanding the data's function in the old system and determining if the function will be the same or different in the new system is of major importance to the Conversion Plan. The structure of the data to be converted can limit the development of the system and affect the choice of software.

Work Product: Develop a Conversion Plan that identifies what conversions are needed and how the conversion(s) will be implemented. Consider the following factors during the development of the Conversion Plan.

- Determine if any portion of the conversion process should be performed manually.
- Determine whether parallel runs of the old and new systems will be necessary during the conversion process.
- Understanding the function of the data in the old system and determining if the use will be the same or different in the new system is important.
- The order that data is processed in the two systems influences the conversion process.
- Volume considerations, such as the size of the database and the amount of data to be converted, influence how the data will be converted. Especially important are the number of reads that are necessary and the time these conversions will take. The Hosting Solution Document provides guidance in assessing database factors.

- User work and delivery schedules, timeframes for reports and end-of-year procedures, and the criticality of the data help determine when data conversion should be scheduled.

- Determine whether data availability and use should be limited during the conversion.

- Plan for the disposition of obsolete or unused data that is not converted.

Review Process: Conduct structured walkthroughs to assure that the Conversion Plan is accurate and complete.

Resource: A Conversion Plan template is available on the MDIT SUITE website.

Activity: **6.6 Develop System Design**

Responsibility: Project Team

Description: The system design is the main technical work product of the System Design Stage. The system design translates requirements into precise descriptions of the components, interfaces, and data necessary before coding and testing can begin. It is a blueprint for the Construction Stage based on the structure and data model established in the Functional Design Stage.

The system design plays a pivotal role in the development and maintenance of a product. The design provides valuable information used by the project manager, quality assurance staff, software configuration management staff, software designers, developers, testers, and maintenance personnel.

The system design is baselined after the system owner's formal approval of the design as described in the System Design Document. Once the system design is baselined, any changes to the design must be managed under change control procedures established in the Software Configuration Management Plan. Approved changes must be incorporated into the System Design Document.

It is important for the system owner/users to understand that some changes to the baselined system design may affect the project scope and therefore can change the project cost, resources, or schedule. It is the responsibility of the project manager and team to identify system owner/user requested changes that would result in a change of project scope; evaluate the potential impact to the project costs, resources, or schedule; and notify the system owner of the project planning revisions that will be required to accommodate their change requests.

Work Product: Major work products include the System Design Document and the updated Requirements Traceability Matrix. Each requirement identified in the Requirements Specification must be traceable to one or more design entities. This traceability ensures that the product will satisfy all of the requirements and will not include inappropriate or extraneous functionality. Revise the Requirements Traceability Matrix developed in the Requirements Definition Stage to relate the system design to the requirements.

Review Process: Conduct a structured walkthrough of the System Design Document and the Requirements Traceability Matrix.

Refer to task 6.6.2, *Conduct System Design Review*, for the system design review process.

Tasks: The following tasks are involved in developing the system design.

6.6.1 Develop System Design Document
6.6.2 Conduct System Design Review

Task: **6.6.1 Develop System Design Document**

Description: The System Design Document records the results of the system design process and describes how the product will be structured to satisfy the requirements identified in the Requirements Specification. The System Design Document is a translation of the requirements into a description of the structure, components, interfaces, and data necessary to support the construction process.

Work Product: Prepare the System Design Document and submit it to the system owner and users for their review and approval. The approved System Design Document is the official agreement and authorization to use the design to build the product. Approval implies that the design is understood, complete, accurate, and ready to be used as the basis for the subsequent lifecycle stages. In other words, once approved this becomes the design baseline. Subsequent changes or additions to the design that receive stakeholder concurrence supersede the existing baseline and establish a new design baseline. Place a copy of the approved System Design Document in the Project File.

Review Process: Conduct structured walkthroughs as needed to ensure that the System Design Document is accurate and complete.

Resource: A System Design Document is available on the MDIT SUITE website.

Task: **6.6.2 Conduct System Design Review**

Description: The System Design Review is a formal technical review of the system design. The purpose of the review is to demonstrate to the system owner and users that the system design can be implemented on the selected platform and accounts for all software and data requirements and accommodates all design constraints (e.g., performance, interface, security, safety, resource, and reliability requirements). The design review should include a review of the validity of algorithms needed to perform critical functions.

Several short System Design Reviews can replace one long review if the product consists of several components that are not highly interdependent. The review process should be a series of presentations by the project team to the system owner and other approval authorities.

Using the System Design Checklist, conduct a System Design Review that demonstrates that the design specifications are capable of supporting the full functionality of the product, as follows:

- All algorithms will perform the required functions.
- The specification is complete, unambiguous and well documented, including timing and sizing, and data and storage allocations.
- The specification is necessary and sufficient for, and directly traceable to, the system design.
- The specification is compatible with every other specification, piece of equipment, facility, and item of system architecture, especially as regards information flow, control, and sequencing.
- The specification is consistent with the abilities of current development and user personnel.

In addition to verifying individual specifications, the System Design Review assesses other project work products to ensure the following.

- The approved design approach is being followed by the team.
- Measures to reduce risk on a technical, cost, and schedule basis are adequate.
- The performance characteristics of the design solution are acceptable.
- Testing will be sufficient to ensure product correctness.

- The resultant application will be maintainable.
- Provisions for automatic, semi-automatic, and manual recovery from hardware/software failures and malfunctions are adequate and documented.
- Diagnostic programs, support equipment, and commercial manuals all comply with the system maintenance concept and specification requirements.

Work Product: Complete the System Design Checklist as part of the System Design review process.

Create and distribute official meeting minutes for each design review session. The minutes should consist of significant questions and answers, action items and individual/group responsible, deviations, conclusions, and recommended courses of action resulting from presentations or discussions. Recommendations that are not accepted should be recorded along with the reason for non-acceptance. Minutes must be distributed to review participants. The system owner determines review performance as follows:

- Approval - The review was satisfactorily completed.
- Contingent Approval - The review is not finished until the satisfactory completion of identified action items.
- Disapproval - The specification is inadequate. Another System Design Review will be required, once the required changes are made to the system design specifications.

Activity: **6.7 Develop Program Specifications**

Responsibility: Project Team

Description: A Program Specification is a written procedural description of each system routine. The Program Specification should provide precise information needed by the developers to develop the code.

Many techniques are available for specifying the system design, such as formal specification languages, program design languages (e.g, pseudo-code or structured English), meta-code, tabular tools (e.g., decision tables), and graphical methods (e.g., flow charts or box diagrams). In object-oriented design, the specification of requirements and preliminary design constraints and dependencies often results in the design language producing the detailed specifications.

Select the technique or combination of techniques that is best suited to the project and to the experience and needs of the developers who will use the system design as their blueprint. The following are suggestions for using the techniques.

- Decision trees are useful for logic verification or moderately complex decisions that result in up to 10-15 actions. Decision trees are also useful for presenting the logic of a decision table to users.

- Decision tables are best used for problems involving complex combinations of up to 5-6 conditions. Decision tables can handle any number of actions; however, large numbers of combinations of conditions can make decision tables unwieldy.

- Structured English is best used wherever the problem involves combining sequences of actions with decisions or loops. Once the main work of physical design has been done and physical files have been defined, it becomes extremely convenient to be able to specify physical program logic using the conventions of structured English, but without getting into the detailed syntax of any particular development language (pseudo-code).

- Standard English is best used for presenting moderately complex logic once the analyst is sure that no ambiguities can arise.

Work Product: Specifications may be produced as documents, graphic representations, formal design languages, records in a database management system, and CASE tool dictionaries. A list of program attributes typically included in a Program Specification is provided at the end of this section.

Review Process: Conduct a series of structured walkthroughs to ensure that the Program Specification is accurate and complete.

Sample Attributes: For each program to be custom built, define the program's functional and technical attributes as they become known. The following is a list of sample program attributes.

- Program identification
- Program name
- Program generic type
- Functional narrative
- Program hierarchical features diagram
- Development dependencies and schedule
- Operating environment
    - equipment
    - development language and version
    - preprocessor
    - operating system
    - storage restrictions
    - security
- Frequency of run
- Data volumes
- Program termination messages
    - normal termination
    - abnormal termination
- Console/printer messages
- Recovery/restart procedures
- Software objectives
- Program input/output diagram
- Data bank information
- Called and calling programs/modules
- Program logic diagrams
- Significant "how-to" instructions
- Telecommunications information

Chapter: **7.0 Construction Stage**

Description: The goal of this stage is to translate the set of technical, computer-oriented system design specifications into a language the computer can understand and execute. Construction involves coding, validation and unit testing by a developer. Any hardware or software procured to support the construction effort is installed. Plans are developed for the installation of the operating environment hardware and software. A training program is designed and a Training Plan that describes the system is produced.

The activities in this stage result in the transformation of the system design into the first complete executable representation of the product. If required, the source code or COTS "glue" code, including suitable comments, is generated using the approved program specifications. The source code is then grouped into processable units and all high-level language units are compiled into object code. Unit testing is performed to determine if the code satisfies the program specifications and is complete, logical, and error free.

The operating documentation is also produced. The operating documentation is required for installing, operating, and supporting the product through its lifecycle.

Input: The following items provide input to this stage:

SEM Templates:

- Conversion Plan
- Functional Design Document
- Maintenance Plan
- Requirements Specification
- Requirements Traceability Matrix
- System Design Document
- Test Plan
- Test Reports

PMM Templates:

- Project Plan
- Quality Management Plan
- Security Plan

Other Inputs:

- Data Dictionary

High-Level Activities:

The remainder of this chapter is divided into sections that describe the specific high-level activities performed during this stage. These activities represent the minimum requirements for a large information systems engineering effort. Notes are provided, as applicable, to assist in customizing these lifecycle stage requirements and to accommodate the different sizes of information systems engineering efforts. The high-level activities are presented in the sections listed below.

7.1 Establish Development Environment
7.2 Develop Programs
7.3 Conduct Unit Testing
7.4 Establish Development Baselines
7.5 Plan Transition to Operational Status
7.6 Generate Operating Documentation
7.7 Develop Training Plan
7.8 Develop Installation Plan

Touch Points:

The following touch points are involved in the Construction Stage:

- Contracts and Procurement
  - Contract Liaison involvement if contract issues arise
- E-Michigan
  - Continue to work with E-Michigan's webmaster, as appropriate, to ensure ADA compliance and Michigan.gov look and feel standards
- Infrastructure Services
  - Infrastructure Specialist involvement in establishing hosting environments
- Security
  - Finalize Network and Data Flow diagrams

Output:

Several work products are produced during this stage. The work products listed below are the minimum requirements for a large project. Deviations in the content and delivery of these work products are determined by the size and complexity of the project. Explanations of the work products are provided under the applicable activities described in the remainder of this chapter.

SEM Templates:

- Conversion Plan (*revised*)
- Installation Plan (*initial*)
- Maintenance Plan (*revised*)
- Requirements Traceability Matrix *(revised)*
- Test Plan (*final*)

- Test Reports (*revised*)
- Training Checklist (*final*)
- Training Plan (*initial*)
- Transition Plan (*initial*)

PMM Templates:

- Project Plan *(revised)*
- Security Plan (*revised*)

Other Outputs:

- Development baselines
- Operating Documentation
  - Users Manual
  - Developer's Reference Manual
- Project Test File
- System units and modules

A diagram showing the work products associated with each high-level activity is provided in *Exhibit 7.0-1, SEM Overview Diagram – Construction Stage Highlighted*.

Review the Project Plan for accuracy and completeness of all Construction Stage activities and make any changes needed to update the information.

Review Process:

Quality reviews are necessary during this stage to validate the product and associated work products. The activities that are appropriate for quality reviews are identified in this chapter and Chapter 2.0, Lifecycle Model. The time and resources needed to conduct the quality reviews should be reflected in the project resources, schedule, and work breakdown structure.

Structured Walkthrough (SWT)

Requirements for a peer review or a more formal structured walkthrough are documented under *Review Process* at the end of each Task, Subtask, or Activity section in this stage. The State of Michigan guide titled *Structured Walkthrough Process Guide* provides a procedure and sample forms that can be used for SWTs. This document is available on the MDIT SUITE website.

Stage Exit

Schedule a Stage Exit as the last activity of the Construction Stage to enable the project approvers to review project deliverables and provide a concur/non-concur position to the project manager. The State of Michigan guide titled *Stage Exit Process Guide* provides a procedure and sample report form that can be used for stage exits. This document is available on the MDIT SUITE website.

References:

Chapter 2.0, Lifecycle Model, *Quality Reviews* provides an overview of the Quality Reviews to be conducted on a project.

Bibliography: The following materials were used in the preparation of the Construction Stage chapter.

1. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Developing Software Lifecycle Processes,* IEEE Std 1074-1991, New York, 1992.

2. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Software User Documentation,* IEEE Std 1063-2001, New York, 2001.

3. U.S. Department of Energy, *Automated Data Processing Systems Development Methodology, Volume 1,* K/CSD/INF/86-3, Vol.1, R3, prepared under contract by Martin Marietta Energy Systems, Inc. at Oak Ridge National Laboratory, August 1987.

4. U.S. Department of Energy, *DOE/NV Software Management Plan,* Nevada Operations Office, May 1991.

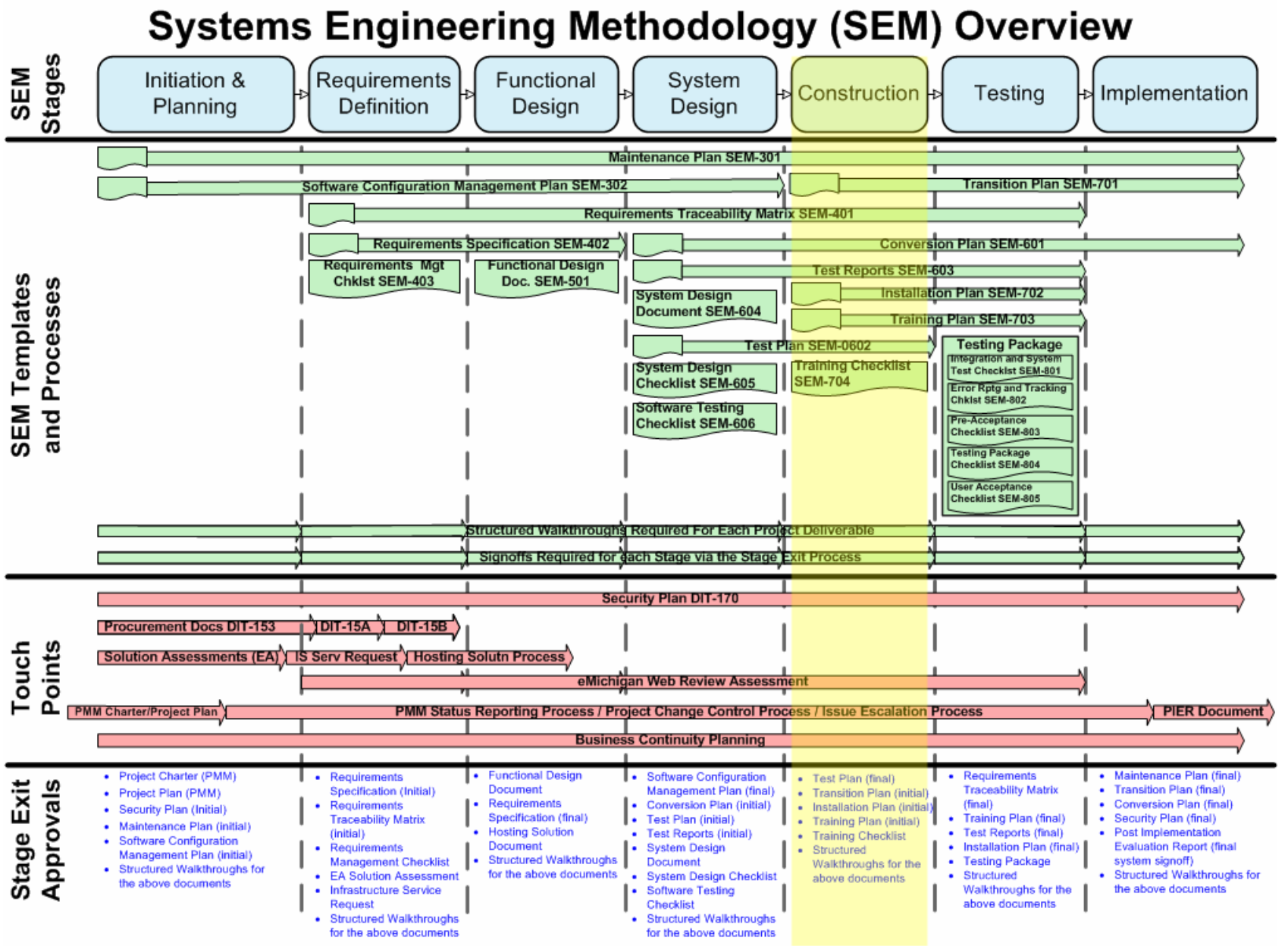
5. U.S. Department of Justice, Immigration and Naturalization Service, *System Development Lifecycle Standards Manual,* 1991.

6. The Institute of Electrical and Electronics Engineers, Inc., *Guide to the Software Engineering Body of Knowledge, Stone Man Trial Version 1.00,* New York, May 2001.

7. Carnegie Mellon University, Software Engineering Institute, *Capability Maturity Model: Guidelines for Improving the Software Process,* Addison Wesley Longman, Inc., 1994.

8. Microsoft Press, *Code Complete, Volume 2,* Michael McConnell, 2004, ISBN: 0-7356-1967-0.

Exhibit 7.0-1 SEM Overview Diagram – Construction Stage Highlighted



Activity: **7.1 Establish Development Environment**

Responsibility: Project Team

Description: Establishing the development environment involves assembling and installing the hardware, software, communications equipment, databases, and other items required to support the coding/construction effort. When the installation of the equipment or software by MDIT Technical Services staff is complete, conduct testing to verify the operating characteristics and functionality of the hardware and software. If required, security software and procedures should be activated when the installations are completed.

If the operational environment is also the development environment, it may be necessary to alter the operational environment to accommodate an infrastructure of purchased hardware and software for use during construction and testing.

Before being integrated into or used to support the product, vendor products should be tested to verify that the product satisfies the following objectives.

- The product performs as advertised/specified.
- The product's performance is acceptable and predictable in the target environment (e.g., testing for LAN certification).
- The product fully or partially satisfies the project requirements.
- The product is compatible with the project team's other hardware and software tools.

Time should be planned for the project team to become familiar with new products. Ensure that the project team members who will use the hardware or software obtain proper training. This may involve attendance at formal training sessions conducted by the vendor or the services of a consultant to provide in-house training.

This is a good time to review the coding practices that were established in the System Design Stage. Make any changes to the standards that are needed to accommodate the procured hardware and software.

Activity: **7.2 Develop Programs**

Responsibility: Project Team Developers

Description: This activity involves generating the source and object code for the product. The code should be written in accordance with the coding practices developed in the System Design Stage. The use of various development languages will dictate whether the code is manually prepared or automatically generated. Regardless of the platform, construction of the code should adhere to a consistent set of coding practices and error prevention procedures. This will promote reliable, maintainable code, developed in the most efficient and cost effective manner.

Code units should be generated in a sequence based on a plan that accounts for factors such as criticality, difficulty, integration and test issues, and needs of customers and users, as appropriate.

For COTS products, although software COTS products attempt to simulate the "plug and play" capability of the hardware world, frequently this is not the case. Most products require some amount of adaptation and integration to work within the State of Michigan environment. The typical solution is to adapt each software COTS product through the use of "wrappers," "bridges," or other "glueware." It is important to note that adaptation does not imply modification of the COTS product. Adaptation can be a complex activity that requires technical expertise at the detailed system and specific COTS component levels. Adaptation and integration must take into account the interactions among custom components, COTS products, any non-developmental item components, any legacy code, and the architecture including infrastructure and middleware elements.

The source and object code should be uniquely identified and stored in a way to facilitate the configuration control measures described in the Software Configuration Management Plan.

Developing programs includes the following tasks.

- Use the Program Specifications developed in the System Design Stage as the basis for the coding effort.
- Generate source code.
- Generate the physical files and database structure.
- Generate video screens and report generation codes.
- Configure predefined options for COTS products.

If conversion of an existing system or data is necessary, generate the program(s) described in the Conversion Plan.

- Conduct preliminary unit testing. When the test output is correct, review the program specification to assure that the unit or module conforms to the specification.

Coding Practices: The following coding practices should be implemented.

- The code development staff should meet at scheduled intervals to discuss problems encountered and to facilitate program integration and uniformity.
- Program uniformity should be achieved by using a standardized set of naming conventions for programs, data elements, variables, and files.
- Modules that can be shared by programs requiring the same functionality should be implemented to facilitate development and maintenance.
- Modules that can be borrowed or reused from other sources that have already been tested should be implemented. Code reuse can lead to faster development.
- Meaningful internal documentation should be included in each program.
- All code should be backed up on a daily basis and stored in an offsite location to avoid catastrophic loss.
- A standard format for storing and reporting elements representing numeric data, dates, times, and information shared by programs should be determined.
- The System Design Document should be updated to reflect any required deviations from the documented design.

Work Products: The following work products are produced.

- Completed units and modules of code.
- Configured COTS options (e.g., databases, tables)
- Test materials generated from preliminary testing.

Each requirement identified in the Requirements Specification must be traceable to the code. This traceability ensures that the product will satisfy all of the requirements and will not include inappropriate or extraneous functionality.

Expand the Requirements Traceability Matrix developed in the Requirements Definition Stage to relate the source and object code to the requirements. Place a copy of the expanded matrix in the Project File.

Review Process: Periodic informal reviews of each developer's work are recommended to keep the project team informed of progress and to facilitate the resolution of any problems that may occur. The combined knowledge and skills of the team members will help to build quality into the product and support the early detection of errors in design, logic, or code.

Conduct structured walkthroughs on the expanded Requirements Traceability Matrix and completed units and modules to assure that the code is accurate, logical, internally well documented, complete, and error free. Structured walkthroughs should also be used to validate that the code is reliable and satisfies the program specifications and project requirements.

For large or complex projects, code inspections may be a more effective method of removing defects and identifying areas where defects may be propagated. Conduct the code inspections at successive stages of code production. Code inspection is a static analysis technique that relies on visual examination of code to detect errors, violations of development standards, and other problems. These inspections are particularly important when code is being produced by several developers or different teams. The inspection team may include experts outside of the project. Ideal times for code inspections occur when code and unit tests are complete, and when the first integration tests are complete. Code inspections should be identified as milestones in the Project Plan.

Activity: **7.3 Conduct Unit Testing**

Responsibility: Project Team Developers

Description: Unit testing is used to verify the input and output for each module. Successful testing indicates the validity of the function or sub-function performed by the module and shows traceability to the design. During unit testing, each module is tested individually and the module interface is verified for consistency with the design specification. All important processing paths through the module are tested for expected results. All error handling paths are also tested.

Unit testing is driven by test cases and test data that are designed to verify requirements, and to exercise all program functions, edits, in-bound and out-bound values, and error conditions identified in the program specifications. If timing is an important characteristic of the module, tests should be generated that measure time critical paths in average and worst-case situations.

Plan and document the inputs and expected outputs for all test cases in advance of the tests. Log all test results. Analyze and correct all errors and retest the unit using the scenarios defined in the test cases. Repeat testing until all errors have been corrected.

While unit testing is generally considered the responsibility of the developer, the project manager or lead developer should be aware of the unit test results.

Work Products: Completion of unit testing for a component signifies internal project delivery of a component or module for integration with other components. Place all components that have completed unit testing under configuration control as described in the Software Configuration Management Plan. These components form the Production Baseline. Configuration controls restrict changes to tested and approved software in the Production Baseline. Subsequent changes or additions to the software that are agreed upon in a System Design Review and receive stakeholder concurrence supersede the existing baseline and establish a new Production Baseline.

Review the draft versions of the Test Plan developed during the System Design Stage. Update the plan, as needed, to reflect any changes made to the design. Deliver the final versions of the Test Plan to the system owner and user for review and approval. Place a copy of the approved plan in the Project File.

Create a Project Test File for all test materials generated throughout the project lifecycle. Place all unit test materials (e.g., inputs, outputs, results and error logs) in the Project Test File. The test cases used for unit testing may become a subset of tests for integration testing.

Review Process: Conduct peer reviews on the test materials to be placed in the Project Test File. Conduct structured walkthroughs on any updated plans (e.g., Integration and System Test Requirements).

Activity: **7.4 Establish Development Baselines**

Responsibility: Project Team Developers

Description: A development baseline is an approved "build" of the product. A build can be a single component or a combination of components. The first development baseline is established after the first build is completed, tested, and approved by the project manager or lead developer. Subsequent versions of a development baseline should also be approved. The approved development baseline for one build supersedes that for its predecessor build.

Conduct internal build tests such as regression, functional, performance, and reliability. Regression tests are designed to verify that capabilities in earlier builds continue to work correctly in subsequent builds. Functional tests focus on verifying that the build meets its functional and data requirements and correctly generates each expected display and report. Performance and reliability tests are used to identify the performance and reliability thresholds of each build.

Once the first development baseline is established, any changes to the baseline must be managed under the change control procedures described in the Software Configuration Management Plan. Approved changes to a development baseline must be incorporated into the next build of the product and revisions made to the affected work products (e.g., Requirements Specification, System Design Document, and Program Specifications).

Work Product: Document the internal build test procedures and results. Identify errors and describe the corrective action that was taken. Place a copy of the internal build test materials in the Project Test File.

Maintain configuration control logs and records as required in the Software Configuration Management Plan.

Expand the Requirements Traceability Matrix developed in the Requirements Definition Stage. All work products developed during the code, unit testing, and build processes must be traced back to the project requirements and system design. This traceability ensures that the product will satisfy all of the requirements and remain within the project scope. Place a copy of the expanded Requirements Traceability Matrix in the Project File.

Review Process: Conduct peer reviews on the internal build test materials to be placed in the Project Test File. Conduct structured walkthroughs on any updated documents (e.g., the Requirements Traceability Matrix).

Activity: **7.5 Plan Transition to Operational Status**

Responsibility: Project Team

Description: Successful transition from acceptance testing to full operational use of the product depends on planning the transition long before the product is installed in its operational environment. In planning for the transition, quantify the operational needs associated with the product and describe the procedures that will be used to perform the transition. Rely on experience and data gathered from previous, similar projects to define these needs.

Work Product: Develop a Transition Plan that describes the detailed plans, procedures, and schedules that will guide the transition process. Coordinate development of the plan with the operational and maintenance personnel. The following issues should be considered in the preparation of a Transition Plan.

- Develop detailed operational scenarios to describe the functions to be performed by the operational support staff, maintenance staff, and users.
- Document the release process. If development is incremental, define the particular process, schedule, and acceptance criteria for each release.
- Describe the development or migration of data, including the transfer or reconstruction of historic data. Schedule ample time for the system owner and user to review the content of reconstructed or migrated data files to reduce the chance of errors or omissions.
- Specify problem identification and resolution procedures for the operational product.
- Define the configuration management procedures that will be used for the operational product. Ideally, the methods defined in the Software Configuration Management Plan that were employed during product development can continue to be used for the operational product.
- Define the scope and nature of support that will be provided by the project team during the transition period.
- Specify the organizations and individuals who will be responsible for each transition activity, ensuring that responsibility for the product by the operations and maintenance personnel increases progressively.
- Identify products and support services that will be needed for day-to-day operations or that will enhance operational effectiveness.

Review Process: Conduct a structured walkthrough to assure that the Transition Plan is logical, accurate, and complete. Involve operational and maintenance personnel in the walkthrough.

Resource: A Transition Plan is available on the MDIT SUITE website.

Activity: **7.6 Generate Operating Documentation**

Responsibility: Project Team/Technical Writer

Description: Plan, organize, and write the operating documentation that describes the functions and features of the product from the users point-of-view. The different ways that users (including system administration and maintenance personnel) will interact with the product must be considered. The needs of the users should dictate the document presentation style and level of detail. Responsibilities for changing and maintaining the documents should be described in each document.

The following are typical operating documents for a large project.

- Users Manual/Online Help Screens
- Developer's Reference Manual
- Systems Administration Manual
- Database Administration Manual
- Operations Manual

It is recommended that a technical writer be involved in the generation of all operating documents. A technical writer works closely with the project team to ensure that documents are grammatically correct; comply with applicable standards; and are consistent, readable, and logical.

Note: The operating documents can be produced as separate manuals or combined to accommodate less complex projects.

Procedure: Use the following procedure to develop the operating documentation.

- Identify the operating documents that need to be developed. Determine if any of the documents can be combined or delivered as multiple volumes.

- Determine whether the documents should be provided as printed material, standalone electronic files, online documentation accessed through the product, or a combination.

- Determine the best presentation method or combination of methods required for each of the documents, such as a traditional manual, quick reference guide or card, or online help.

- Identify all of the features of the user interface and the tasks users will perform.

- Identify the users' needs and experience levels to determine:

  - The amount of user interaction, level of interaction, and whether the interaction is direct or indirect.

  - The appropriate level of detail (e.g., the Users Manual should not contain highly technical terms and explanations that may confuse or frustrate a user).

- Determine the document content and organization based on whether the document will be used more as an instructional tool or a reference guide.

- Develop descriptions of each function and feature of the product and organize the information to facilitate quick, random access.

- Provide appropriate illustrations and examples to enhance clarity and understanding.

- Establish a schedule for the documents to be reviewed after the product goes into production. Operating documents must be kept up-to-date as long as the product remains in production.

Work Products: Refer to each of the following tasks for applicable work products.

Review Process: Refer to each of the following tasks for applicable review processes.

Tasks: The following tasks describe the minimum requirements for operating documentation.

7.6.1 Produce Users Manual
7.6.2 Produce Developer's Reference Manual

Task: **7.6.1 Produce Users Manual**

Description: The Users Manual provides detailed information users need to access, navigate through, and operate the product. Users rely on the Users Manual to learn about the product or to refresh their memory about specific functions. A Users Manual that is organized functionally so that the information is presented the same way the product works helps users understand the flow of menus and options to reach the desired functions.

Different categories of users may require different types of information. A modular approach to developing the Users Manual to accommodate the needs of different types of users eliminates duplication and minimizes the potential for error or omission during an amendment or update. For example, separate general information that applies to all users from the special information that applies to selected users such as system administrators or database administrators. The special information can be presented in appendixes or supplements that are only provided to the users who need the information.

Work Product: Write the draft Users Manual in clear, non-technical terminology that is oriented to the experience levels and needs of the user(s). The following are typical features of a users manual.

- Overview information on the history and background of the project and the architecture, operating environment, and current version or release of the product.
- Instructions for how to install, setup, or access the product.
- Complete coverage of all functions, presented in a logical, hierarchical order.
- Accurate pictures of screens and reports, ideally with data values shown, so the user can easily relate to examples.
- In-depth examples and explanations of the areas of the product that are most difficult to understand.
- Clear delineation of which features are accessible only to specific users.
- Instructions on accessing and using online help features.
- Procedures for data entry.

- Descriptions of error conditions, explanations of error messages, and instructions for correcting problems and returning to the function being performed when the error occurred.

- Instructions for performing queries and generating reports.

- Who to contact for help or further information.

Note: For large or complex products, separate manuals (e.g., User's Manual, Database Administrator's Manual, and System Administrator's Manual) may be necessary to address the needs of different categories of users.

For very small projects, a quick reference guide or card may be more appropriate than a full-scale Users Manual. The guide or card should be designed to provide a quick reference of logon, logoff, and commands for frequently used functions.

For projects of any size, a quick reference card may be developed as a supplement to more detailed user documentation.

Review Process: Conduct structured walkthroughs for the draft Users Manual or set of user documents to assure that the documentation is complete, easy to use, and accurately reflects the product and its functions.

The draft user documentation will be tested and verified with the product during the Testing Stage.

Task: **7.6.2 Produce Developer's Reference Manual**

Description: The Developer's Reference Manual contains information about program development used by the maintenance staff to maintain the programs, databases, interfaces, and operating environment. The Developer's Reference Manual should provide an overall conceptual understanding of how the product is constructed and the details necessary to implement corrections, changes, or enhancements.

The Developer's Reference Manual describes the logic used in developing the product and the functional and system flow to help the maintenance staff understand how the programs fit together. The information should enable a developer to determine which programs may need to be modified to change a system function or to fix an error.

Work Product: The following are typical features of a Developer's Reference Manual.

- A description of the technical environment, including versions of the development language(s) and other proprietary software packages.
- A brief description of the design features including descriptions of unusual conditions and constraints.
- An overview of the architecture, program structure, and program calling hierarchy.
- The design and coding practices and techniques used to develop the product.
- Concise descriptions of the purpose and approach used for each program.
- Layouts for all data structures and files used in the product.
- Descriptions of maintenance procedures, including configuration management, program checkout, and system build routines.
- The instructions necessary to compile, link, edit, and execute all programs.
- Manual and automated backup procedures.
- Error-processing features.

Use appendixes to provide detailed information that is likely to change as the product is maintained. For example, a list of program names and a synopsis of each program could be included as an appendix.

Review Process: Conduct structured walkthroughs of the draft Developer's Reference Manual to assure that the documentation is complete, easy to use, and accurately reflects the product and its functions.

The draft Developer's Reference Manual will be tested and verified with the product during the Testing Stage.

Activity: **7.7 Develop Training Plan**

Responsibility: Project Team

Description: A Training Plan defines the training needed to implement and operate the product successfully. The Training Plan should address the training that will be provided to the system owner, users, and maintenance staff. When new hardware or software is being used, affected personnel will need hands-on experience before bringing the new system (equipment and/or software) into daily operation.

Training must address both the knowledge and the skills required to operate and use the system effectively. Design the training to accomplish the following objectives.

- Provide trainees with the specific knowledge and skills necessary to perform their work.
- Prepare training materials that will sell the product as well as instruct the trainees. The training should leave the trainees with the enthusiasm and desire to use the new product.
- Account for the knowledge and skills the trainees bring with them, and use this information as a transition to learning new material.
- Anticipate the needs for follow-on training after the product is fully operational, including refresher courses, advanced training, and repeats of basic courses for new personnel.
- Build in the capability to update the training as the product evolves.

Involve the system owner and key users in the planning to determine the education and training needs for all categories of users (managers, users, and maintenance staff).

The Training Plan should address the following issues:

- Identify the organization's training policy for meeting training needs.
- Ensure software managers have received orientation on the training.
- Ensure training courses prepared at the organization level are developed and maintained according to organizational standards.

- Ensure a procedure for required training is established and used to determine whether individuals already possess the knowledge and skills required to perform in their designated area.
- Ensure measurements are made and used to determine the status of training activities.
- Ensure that training activities are reviewed with senior management on a periodic basis.
- Ensure the training is independently evaluated on a periodic basis for consistency with, and relevance to, the organization's needs.
- Ensure the training activities and work products are reviewed and/or audited and the results are reported.
- Ensure training records are properly maintained.

Work Product: Prepare a draft Training Plan that describes the training and at a minimum addresses the following issues.

- Identifies personnel to be trained. Review the list of trainees with the system owner and users to ensure that all personnel who should receive training have been identified.
- Defines the overall approach to training and the required training courses.
- Establishes the scope of the training needed for users, management, operations, and maintenance personnel.
- Defines how and when training will be conducted. Specify instructor qualifications, learning objectives, and mastery or certification requirements (if applicable).
- Identifies any skill areas for which certification is necessary or desirable. Tailor the training to the certification requirements.
- Establishes a preliminary schedule for the training courses. The schedule must reflect training requirements and constraints outside the project. Schedule individual courses to accommodate personnel who may require training in more than one area. Identify critical paths in the training schedule such as the time period for the product's installation and conversion to production status.

- Defines the required course(s), outlines their content and sequence, and establishes training milestones to meet transition schedules.

- Tailors the instruction methods to the type of material being presented. Include classroom presentation, interactive computer-assisted instruction, demonstrations, individual video presentations, and hands-on experience, either live or simulated.

- Identifies trainers who are technically knowledgeable and were involved in the design and development of the system. For projects with extensive and formal training requirements, it may be necessary to provide training for the trainers.

- Consider availability of the following: users, system-tested software, training rooms and equipment, and the completion of system documentation and training materials.

Complete the Training Checklist to ensure that all activities and work products are complete.

Place a copy of the initial Training Plan and completed Training Checklist in the Project File. The plan will be reviewed and updated during the Testing Stage.

Review Process: Conduct a structured walkthrough to assure that the initial Training Plan is accurate and complete.

Resource: The Training Plan template and Training Checklist are available on the MDIT SUITE website.

Activity: **7.8 Develop Installation Plan**

Responsibility: Project Team

Description: The Installation Plan is prepared to specify the requirements and procedures for the full-scale installation of the developed product at the system owner's and all users' work sites. The plan also addresses the installation of any hardware, off-the-shelf software, firmware, and communications equipment needed to operate the product at each site. In developing an Installation Plan consider each site's requirements for continuity of operations, level of service, and the needs of the project team, users, maintenance personnel, and management.

Work Product: Work closely with the system owner and representatives from the user sites to assure that all site-specific hardware, software, and communications installation requirements are addressed in the Installation Plan. Develop an initial Installation Plan that addresses the following issues.

- Schedule of all installation activities.
- Items to be delivered to each installation site.
- Number and qualifications of personnel performing installation.
- Equipment environmental needs and installation instructions.
- Hardware, software, firmware, tools, documentation, and space required for each installation.
- Special requirements governing the movement of equipment to each site.
- Communications requirements.
- Dependencies among activities affected by installation.
- Installation tests to assure the integrity and quality of the installed product.

Ensure any special requirements regarding, e.g., network resources and web-based applications are adequately documented. Place a copy of the initial Installation Plan in the Project File.

Review Process: Conduct a structured walkthrough to assure that the initial Installation Plan is accurate and complete. The Installation Plan will be reviewed and revised as needed during the Testing and Implementation Stages.

Resource: An Installation Plan template is available on the MDIT SUITE website.

Chapter: **8.0 Testing Stage**

Description: Testing activities focus on interfaces between and among components of the product, such as functional correctness, system stability, overall system operability, system security, privacy and sensitive information control, and system performance requirements (e.g., reliability, maintainability, and availability). Testing performed incrementally provides feedback on quality, errors, and design weaknesses early in the integration process.

In this stage, components are integrated and tested to determine whether the product meets predetermined functionality, performance, quality, interface, and security requirements. Once the product is fully integrated, system testing is conducted to validate that the product will operate in its intended environment, satisfies all user requirements, and is supported with complete and accurate operating documentation. User Acceptance Testing (UAT) follows System Testing, and solicits feedback from users to make any final adjustments to the programming before releasing the product for implementation.

Input: The following items provide input to this stage:

SEM Templates:

- Conversion Plan
- Installation Plan
- Maintenance Plan
- Requirements Specification
- Requirements Traceability Matrix
- Test Plan
    - Integration testing (component to component)
    - Performance testing (load, stress, etc.)
    - System testing (end to end)
    - User acceptance testing (UAT)
- Test Reports
    - Integration test reports
    - Performance test report
    - System test reports
    - User Acceptance test reports
- Transition Plan
- Training Plan

PMM Templates:

- Project Plan
- Quality Management Plan
- Security Plan

Other Inputs:

- Development Baselines
- Operating Documentation
  - Users Manual
  - Developer's Reference Manual
- Project Test File
- Software Modules

High-Level Activities:

The remainder of this chapter is divided into sections that describe specific high-level activities performed during this stage. These activities represent the minimum requirements for a large information systems engineering effort. *Notes* are provided, as applicable, to assist in customizing these lifecycle stage requirements to accommodate the different sizes of information systems engineering efforts. The high-level activities are presented in the sections listed below.

8.1 Conduct Integration Testing
8.2 Conduct System Testing
8.3 Conduct User Acceptance Testing

Touch Points:

The following touch points are involved in the Testing Stage:

- Contracts and Procurement
  - Contract Liaison involvement if contract issues arise
- E-Michigan
  - Continue to work with E-Michigan's webmaster, as appropriate, to ensure ADA compliance and Michigan.gov look and feel standards
- Infrastructure Services
  - Infrastructure Specialist involvement as documented in the Infrastructure Services Request (ISR)
- Security
  - Include application testing for security controls

Output:

Several work products are produced during this stage. The work products listed below are the minimum requirements for a large project. Deviations in the content and delivery of these work products are determined by the size and complexity of the project. Explanations of the work products are provided under the applicable activities described in the remainder of this chapter.

SEM Templates:

- Error Reporting and Tracking Checklist (*final*)
- Integration and System Testing Checklist (*final*)
- Pre-acceptance Checklist (*final*)
- Testing Package Checklist (*final*)
- User Acceptance Checklist (*final*)
- Conversion Plan *(revised, if needed)*
- Installation Plan *(final)*
- Maintenance Plan *(revised)*
- Requirements Traceability Matrix *(final)*
- Test Reports *(final)*
    - Integration test reports
    - Performance test report
    - System test reports
    - User Acceptance test reports
- Training Plan *(final)*
- Transition Plan *(revised)*

PMM Templates:

- Project Plan *(revised)*
- Security Plan *(revised)*

Other Outputs:

- Operating Documents *(final)*
    - Users Manual
    - Developer's Reference Manual

A diagram showing the work products associated with each SEM stage is provided in *Exhibit 8.0-1, SEM Overview*. The activities for this stage are emphasized in bold.

Review the Project Plan for accuracy and completeness of all Testing Stage activities and make any changes needed to update the information.

Review Process: Quality reviews are necessary during this stage to validate the product and associated work products. The activities that are appropriate for quality reviews are identified in this chapter and Chapter 2.0, Lifecycle Model. The time and resources needed to conduct the quality reviews should be reflected in the project resources, schedule, and work breakdown structure.

Structured Walkthrough (SWT)
Requirements for a peer review or a more formal structured walkthrough are documented under *Review Process* at the end of each Task, Subtask, or Activity

section in this stage. The State of Michigan guide titled *Structured Walkthrough Process Guide* provides a procedure and sample forms that can be used for SWTs. This document is available on the MDIT SUITE website.

Stage Exit

Schedule a Stage Exit as the last activity of the Testing Stage to enable the project approvers to review project deliverables and provide a concur/non-concur position to the project manager. The State of Michigan guide titled *Stage Exit Process Guide* provides a procedure and sample report form that can be used for stage exits. This document is available on the MDIT SUITE website.

References:

Chapter 2.0, Lifecycle Model, *Quality Reviews* provides an overview of the Quality Reviews to be conducted on a project.

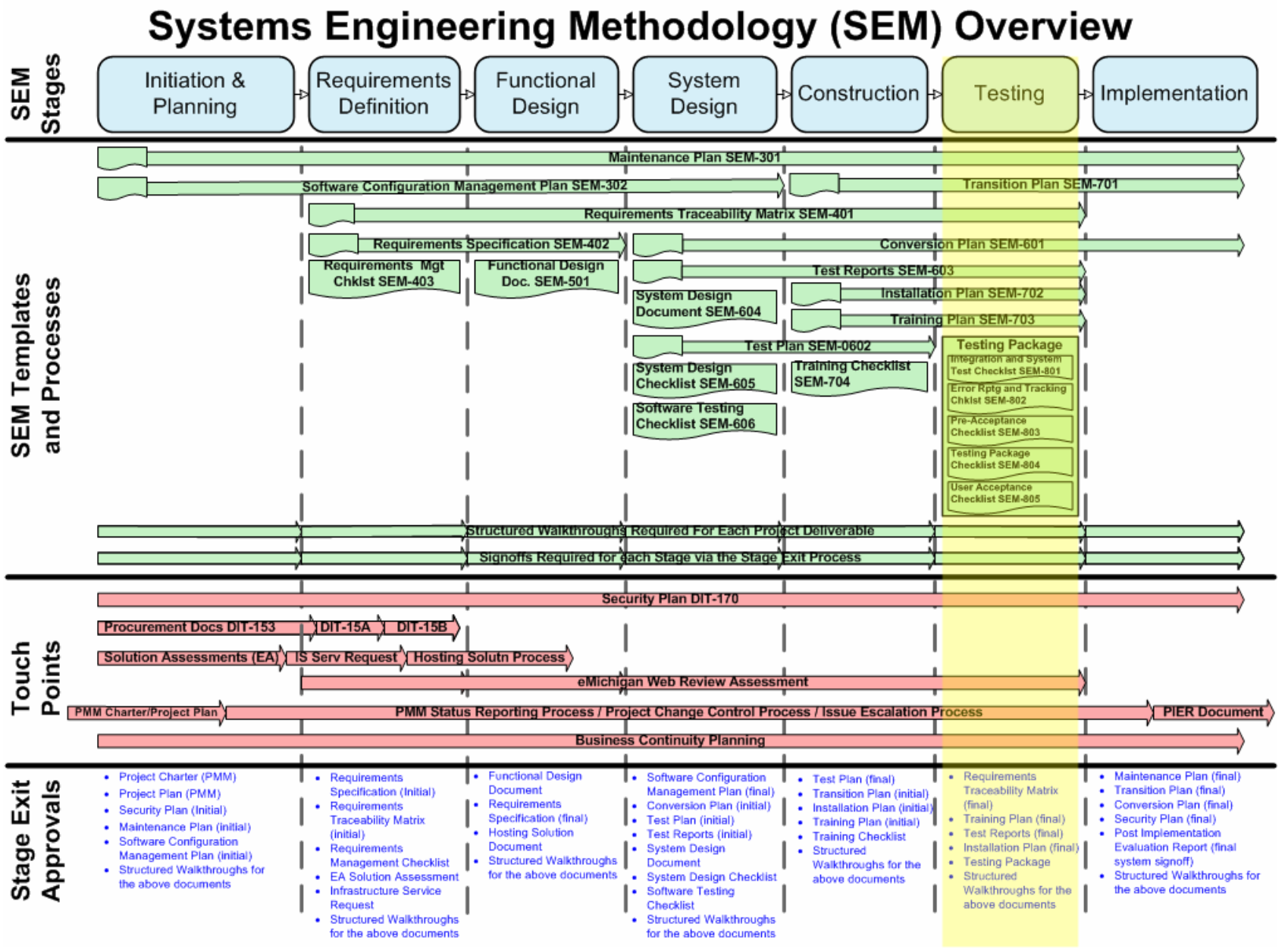
Bibliography:

The following materials were used in the preparation of the Testing Stage chapter.

1. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Developing Software Lifecycle Processes,* IEEE Std 1074-1991, New York, 1992.

2. U.S. Department of Commerce, National Institute of Standards and Technology, *Guide to Software Acceptance,* 500-180, Washington, D.C., 1990.

Exhibit 8.0-1 SEM Overview Diagram – Testing Stage Highlighted



Activity: **8.1 Conduct Integration Testing**

Responsibility: Project Team Developers

Description: Integration testing is the first activity in the Testing Stage and requires special attention to preparation. The Pre-Acceptance Checklist, Integration and System Test Checklist, and Testing Package Checklist each provide the necessary steps for their preparation. These documents are available on the MDIT SUITE website.

During integration, the components constructed by the development staff, off-the-shelf software purchased from vendors, and reusable code or modules obtained from other sources are assembled into one product. Each assembly is tested in a systematic manner in accordance with the Integration Section of the Test Plan. An incremental approach to integration enables verification that as each new component is integrated, it continues to function as designed and both the component and the integrated product satisfy their assigned requirements.

Integration testing is a formal procedure that must be carefully planned and coordinated with the completion dates of the unit-tested modules. Integration testing begins with a structure where called sub-elements are simulated by stubs. A stub is a simplified program or dummy module designed to provide the response (or one of the responses) that would be provided by the real sub-element. A stub allows testing of calling program control and interface correctness. Stubs are replaced by unit-tested modules or builds as integration testing proceeds. This process continues one element at a time until the entire system has been integrated and tested.

Integration testing may be performed using "bottom up" or "top down" techniques. Most integration test plans make use of both bottom-up and top-down techniques. Scheduling constraints and the need for parallel testing will affect the test approach.

The bottom-up approach incorporates one or more modules into a build; tests the build; and then integrates the build into the structure. The build normally comprises a set of modules that perform a major function of the system. Initially, the function may be represented by a stub that is replaced when the build is integrated.

In the top-down approach, individual stubs are replaced so that the top-level control is tested first, followed by stub replacements that move downward in the structure. Using top-down integration, all modules that comprise a major function are integrated, thereby allowing an operational function to be demonstrated prior to completion of the entire system.

Given the incremental nature of the Testing Stage, completion and sign-off of the Integration Section of the Integration and System Testing Checklist is required prior to moving on to System Testing.

Work Products: Each requirement identified in the Requirements Specification must be tested during integration testing. This traceability ensures that the product will satisfy all of the requirements and will not include inappropriate or extraneous functionality. Expand the Requirements Traceability Matrix developed in the Requirements Definition Stage to relate the integration test to the requirements. Place a copy of the expanded matrix in the Project File.

At the completion of each level of integration testing, a test report is written. The report documents test results and lists any discrepancies that must be resolved before the tested components can be used as the foundation for another integration level. Place a copy of all integration test materials in the Project Test File.

A final test report is generated at the completion of integration testing indicating any unresolved difficulties that require management attention. Place a copy of the final Integration Test Report in the Project File.

Sign-off of the Integration section of the Integration and System Checklist signifies completion of the Integration Testing activities.

A formal reporting system by which detected errors and discrepancies are recorded and fully described is recommended. These reports will help to confirm that all known errors are fixed before delivery of the completed product. Error reports also help to trace multiple instances of the same error or anomalous behavior, so that error correction and prevention assignments can be implemented. The Quality Assurance representative assigned to the project can provide assistance in developing and using an error reporting/tracking system. An Error Reporting and Tracking Checklist document is available on the MDIT SUITE website.

Review Process: Conduct a structured walkthrough of the Requirements Traceability Matrix and final Test Report based on type of testing (regression, performance, integration, etc.).

Activity: **8.2 Conduct System Testing**

Responsibility: Project Team or Independent Test Team

Description: During system testing, the completely integrated product is tested to validate that the product meets all requirements. System response timing, memory, performance, security, and the functional accuracy of logic and numerical calculations are verified under both normal and high-load conditions. Query and report capabilities are exercised and validated. All operating documents are verified for completeness and accuracy.

System testing is conducted on the system test bed using the methodology and test cases described in the System Test Requirements section of the Requirements Specification document. The system test environment should be as close as possible to the actual production system environment. Either the project team or an independent test team conducts system testing to assure that the system performs as expected and that each function executes without error. The results of each test are recorded and upon completion included as part of the project test documentation.

Note that regression testing is a critical aspect of system testing. It is performed in order to verify that system modifications have not caused unintended effects and that the software or system component still complies with its specified requirements.

When errors are discovered, they should be reviewed by the test team leader to determine the severity and necessary subsequent action. If appropriate, minor problems can be corrected and regression tested by the project team developers within the time frame allotted for the system test. Any corrections or changes to the product must be controlled under configuration management. Major problems may be cause to suspend or terminate the system test, which should then be rescheduled to begin after all of the problems are resolved.

Users may be encouraged to participate in the system tests to gain their confidence in the product and to receive an early indication of any problems from the user's perspective. Inform users that errors and discrepancies may occur during testing and explain the error correction, configuration management, and retest processes.

Given the incremental nature of the Testing Stage, completion and sign-off of the Integration and System Testing Checklist is required prior to moving on to User Acceptance Testing.

Work Products: Review the draft versions of the operating documents, Training Plan, and Installation Plan. Update the documents as needed. Deliver the final versions of the operating documents, Training Plan, and Installation Plan to the system owner and user for review and approval. Place a copy of the approved documents in the Project File.

Place a copy of all system test materials (e.g., inputs, outputs, results, and error logs) in the Project Test File.

Sign-off of the Integration and System Testing Checklist and the Pre-Acceptance Checklist signifies completion of the System Testing activities.

Generate a test report at the conclusion of the system test process. The report documents the system test results and lists any discrepancies that must be resolved before the software product is ready for acceptance testing. Place a copy of the report in the Project File.

Review Process: Conduct a structured walkthrough of the system test report to ensure that all areas of System Testing are complete and that all issues have been resolved.

Activity: **8.3 Conduct User Acceptance Testing**

Responsibility: Project Team and Acceptance Test Team

Description: Acceptance of a delivered product is the ultimate objective of a development project. Acceptance testing is used to demonstrate the product's compliance with the system owner's requirements and acceptance criteria.

At the system owner's discretion, acceptance testing may be performed by the project team, by the system owner and users with support from the project team, or by an independent verification and validation team. Whenever possible, users should participate in acceptance testing to assure that the product meets the users' needs and expectations. All acceptance test activities should be coordinated with the system owner, user(s), operations staff, and other affected organizations.

Acceptance testing is conducted in the test environment using acceptance test data and test procedures established in the Acceptance Test Requirements section of the Requirements Specification. Testing is designed to determine whether the product meets functional, performance, and operational requirements. If acceptance testing is conducted on an incremental release basis, the testing for each release should focus on the capabilities of the new release while verifying the correct operation of the requirements incorporated in the previous release.

If the project team is not conducting the User Acceptance Test (UAT), training may be required for the personnel performing the testing. The acceptance test participants and their experience with the product and the operating environment should have been identified in the Acceptance Test Requirements within the Requirements Specification.

Acceptance testing usually covers the same requirements as the system test. Acceptance testing may cover additional requirements that are unique to the operational environment. The results of each test should be recorded and included as part of the project test documentation.

UAT is typically the final phase in a software development process in which the software is given to the intended audience to be tested for functionality. UAT is done by making the software available for testing by an in-house testing panel comprised of users who would be using the product in real-world applications. UAT is done in order to get feedback from users to make any final adjustments to the programming before releasing the product to the intended user community.

The level of training will depend on the testers' familiarity with the product and the platform on which the product will run. The advantage of having users acceptance test the product is that they are the experts most familiar with the business information flow and how the product must fit into the workplace.

It is recommended that the operating documents and other test materials be distributed to the test team prior to the actual start of the acceptance test training. This will give the test team time to become familiar with the product and the test process and procedures.

Subject the test environment to strict, formal configuration control to maintain the stability of the test environment and to assure the validity of all tests. Review the acceptance test environment, including the test procedures and their sequence, with the system owner and user before starting any tests.

Testing is complete when all tests have been executed correctly. If one or more tests fail, problems are documented, corrected, and retested. If the failure is significant, the acceptance test process may be halted until the problem is corrected.

Completion and sign-off of User Acceptance Testing is required prior to moving on to the Implementation Stage.

Work Products: Sign-off of the User Acceptance Checklist and the Testing Package Checklist signifies completion of the Testing Stage.

Prepare a formal Acceptance Test Report. Summarize the test procedures executed, any problems detected and corrected, and the projected schedule for correcting any open problem reports. Place a copy of all acceptance test materials in the Project Test File.

Review Process: A Readiness Review is a combined quality assurance and configuration management activity. It focuses on the results of the acceptance test and the readiness of the product to go into production including review of security, sensitive information and privacy control. Descriptions of the components of a Readiness Review are included below.

It includes a functional configuration audit to determine whether the test records demonstrate that the product meets its technical requirements and a physical configuration audit to determine whether the product technical documentation is complete and accurately describes the product.

The Functional Control Audit (FCA) compares the system's elements (programs/modules) to the requirements documented in the current version of the Requirements Specification to assure that the modification addresses all, and only, those requirements. The results of the FCA should be documented, identifying all discrepancies found, and the plans for their resolution.

The Physical Control Audit (PCA) compares the components (programs/modules) with its supporting documentation to assure that the documentation to be delivered correctly describes the system components. All discrepancies noted during the PCA, along with plans for their resolution, should be documented.

During the Readiness Review examine acceptance test results with the system owner and user. Document any problems, determine solutions to the problems, and implement action plans. Once any problems associated with the acceptance test are resolved, the product is ready for formal acceptance by the system owner.

A successful Readiness Review establishes the operational baseline for the product. The operational baseline is the final baseline. It consists of the product and the technical documentation that describes the operational product and its characteristics. It contains the current functional baseline, the product baselines for the configuration items comprising the system, and other system-level technical documentation generated during the lifecycle.

If the operational product requires enhancements or changes to correct problems, each new release should be preceded by a Readiness Review, after which the updated system documentation is established as a new operational baseline superseding the previous one.

Chapter: **9.0 Implementation Stage**

Description: Implementation of the product is initiated after all application testing has been successfully completed. This stage involves the activities required to install the software, databases, or data that comprise the product onto the hardware platform at the site(s) of operation. The activities associated with this stage should be performed each time the product is installed at a production site.

User training may be required to complete the implementation process. A description of the training necessary for developers, testers, users, and operations staff is provided in the Training Plan.

Input: The following items provide input to this stage:

SEM Templates:

- Conversion Plan
- Installation Plan
- Maintenance Plan
- Training Plan
- Transition Plan

PMM Templates:

- Project Plan
- Quality Management Plan
- Security Plan

Other Inputs:

- Operating Documents
  - Users Manual
  - Developer's Reference Manual

High-Level Activities: The remainder of this chapter is divided into sections that describe the specific high-level activities performed during this stage. These activities represent the minimum requirements for a large information systems engineering effort. *Notes* are provided, as applicable, to assist in customizing these lifecycle stage requirements to accommodate the different sizes of information systems engineering efforts. The high-level activities are presented in the sections listed below.

9.1 Perform Installation Activities
9.2 Conduct Installation Tests
9.3 Transition to Operational Status

Touch Points: The following touch points are involved in the Implementation Stage:

- Contracts and Procurement
  - Contract Liaison involvement to close out all open contracts and/or purchase orders related to this systems development effort
- Security
  - Work with the MDIT Security Liaison to finalize and get final signoff of the Security Plan (DIT-0170)
- Other
  - Finalize the Business Continuity Planning process.

Output: Several work products are produced during this stage. The work products listed below are the minimum requirements for a large project. Deviations in the content and delivery of these work products are determined by the size and complexity of the project. Explanations of the work products are provided under the applicable activities described in the remainder of this chapter.

SEM Templates:

- Conversion Plan (*final*)
- Maintenance Plan *(final)*
- Transition Plan (*final)*

PMM Templates:

- Project Plan *(final)*
- Post Implementation Evaluation Report (PIER) (*final*)
- Security Plan (*final*)

Other Outputs:

- Converted data or system files
- Installation Test materials
- Operating documents
- Operational software product

A diagram showing the work products associated with each SEM stage is provided in *Exhibit 9.0-1, SEM Overview Diagram.* The activities for this stage are emphasized in bold.

Review the Project Plan for accuracy and completeness of all Implementation Stage activities and make any changes needed to update the information.

Review Process: Quality reviews are necessary during this stage to validate the product and associated work products. The activities that are appropriate for quality reviews are identified in this chapter and Chapter 2.0, Lifecycle Model. The time and resources needed to conduct the quality reviews should be reflected in the project

resources, schedule, and work breakdown structure.

Structured Walkthrough (SWT)
Requirements for a peer review or a more formal structured walkthrough are documented under *Review Process*, at the end of each Task, Subtask, or Activity section in this stage. The State of Michigan guide titled *Structured Walkthrough Process Guide* provides a procedure and sample forms that can be used for SWTs. This document is available on the MDIT SUITE website.

Stage Exit
Schedule a Stage Exit as the last activity of the Implementation Stage to enable the project approvers to review project deliverables and provide a concur/non-concur position to the project manager. The State of Michigan guide titled *Stage Exit Process Guide* provides a procedure and sample forms that can be used for Stage Exits. This document is available on the MDIT SUITE website.

References: Chapter 2.0, Lifecycle Model, *Quality Reviews* provides an overview of the Quality Reviews to be conducted on a project.

Bibliography: The following materials were used in the preparation of the Installation and Acceptance Stage chapter.

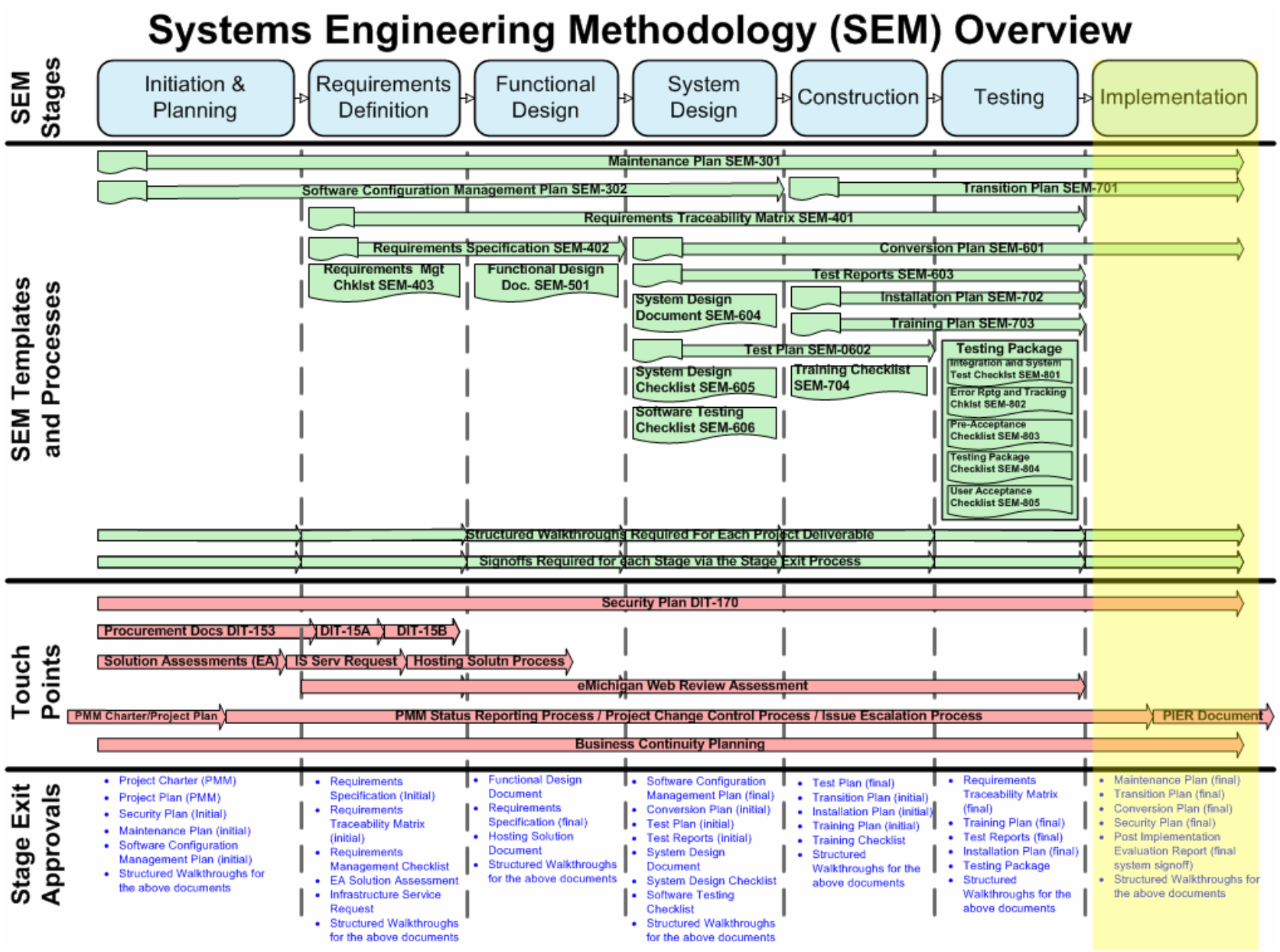
1. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Developing Software Lifecycle Processes,* IEEE Std 1074-1991, New York, 1992.

2. U.S. Department of Commerce, National Institute of Standards and Technology, *Guide to Software Acceptance,* 500-180, Washington, D.C., 1990.

3. U.S. Department of Energy, *Automated Data Processing Systems Development Methodology, Volume 1,* K/CSD/INF/86-3, Vol.1, R3, prepared under contract by Martin Marietta Energy Systems, Inc, at Oak Ridge National Laboratory, August 1987.

4. U.S. Department of Energy, *DOE/NV Software Management Plan,* Nevada Operations Office, May 1991.

Exhibit 9.0-1 SEM Overview Diagram – Implementation Stage Highlighted



Activity: **9.1 Perform Installation Activities**

Responsibility: Project Team

Description: The installation process involves loading, copying, or migrating the software and data, if required, to the production platform and the provision of operating documentation and other support materials at each site. The installation of firmware, hardware, and communications equipment may also be involved.

If a current system exists, implement system and data conversion in accordance with the procedures described in the Conversion Plan. Each data and file conversion should include a confirmation of data and file integrity. Determine what the output in the new product should be compared with the current system, and assure that the data and files are synchronized.

At each installation site, inspect the facility to assure that site preparation is complete and in accordance with the Installation Plan. Initiate any actions that are needed to complete the preparations. Conduct an inventory of all vendor provided hardware, software, firmware, and communications equipment.

Follow the procedure specified in the Installation Plan when installing the software, hardware, and other equipment. Monitor all installation activities including those performed by vendors.

Procedure: Use the following procedure to perform the installation activities.

- Coordinate the installation with the system owner, users, operations staff, and other affected organizations.
- Ensure that any necessary modifications to the physical installation environment are completed.
- Inventory and test the hardware that will support the product. This inventory should be performed in advance of the planned installation date to allow time for missing hardware to be obtained and malfunctioning equipment to be replaced or repaired.
- If the product requires an initial data load or data conversion, install and execute the tested programs to perform this process.
- Install the software product onto the hardware platform.

Activity: **9.2 Conduct Installation Tests**

Responsibility: Project Team

Description: Ensure the integrity and quality of the installed product by executing the installation tests defined in the Installation Plan. Testing is performed to verify that the product has been properly installed and is fully operational and in production.

The installation test(s) are designed to validate all functions of the product and should specify a standard set of test results and tolerances. If the product being installed is a modification to an existing system, all remaining functions and code that may be affected by the new product should be tested.

Document any problems and identify corrective action. Select a diagnostic package that will pinpoint problems quickly and allow for timely corrections. Retest all equipment and software after a repair, replacement, or modification.

Certify each component on successful completion of installation and checkout. When installation is complete, rerun a portion or all of the system test and dry-run the acceptance test procedures to verify correct operation of the product.

Conduct installation testing to verify the following:

- Security functions
- Integration with the current software
- Interfaces with other systems
- System functionality based on the requirements

Work Product: Place a copy of all Installation Test materials in the Project File.

Review Process: Conduct structured walkthroughs of the Installation Test materials.

Activity: **9.3 Transition to Operational Status**

Responsibility: Transition Team

Description: The transition of the product to full operational status begins after the formal acceptance by the system owner. Use the procedures described in the Transition Plan to implement the transition processes. Conduct or support stress tests and other operational tests. Determine product tolerances to adverse conditions, failure modes, recovery methods, and specification margins. Complete any training and certification activities. Ensure that support to be provided by contractors begins as planned.

The project team is usually expected to provide operational and technical support during the transition. Identify project team personnel with a comprehensive understanding of the product who can provide assistance in the areas of installation and maintenance, test, and documentation of changes. Technical support may involve the analysis of problems in components and operational procedures, the analysis of potential enhancements, and vendor-supplied upgrades to components (such as the operating system or database management system).

Transition to full operational status should be an event-oriented process that is not complete until all transition activities have been successfully performed. Withdraw the support of the project team personnel in a gradual sequence to ensure the smooth operation of, and user confidence in, the product. At the conclusion of the transition process, plan a formal transfer of all responsibility to the maintenance staff. This includes working with DIT Technical Services to add the application to the Configuration Management Data Base (CMDB) Application Board.

All Project File materials, operating documents, a list of any planned enhancements, and other pertinent records should be turned over to the maintenance staff. Access rules must be modified to provide access to the product by the maintenance staff and to remove access by the project team and other temporary user accesses. Programs, files, and other support software should be in the production library and deleted from the test library, where appropriate.

For major systems involving multiple organizations and interfaces with other systems, a formal announcement of the transition to production is recommended. The announcement should be distributed to all affected groups. The names and telephone numbers of the maintenance staff should be included.

Work Product: The system is transitioned into operational status. Project File materials, operating documents, and other pertinent records are turned over to the maintenance staff.

Review Process: All reviews related to the functionality were completed prior to the system being placed into operational status.

Page inserted for consistency in section start points.

Appendices

Table of Contents

Page inserted for consistency in section start points.

Appendix A – Systems Engineering Methodology Glossary

Key Terms and Acronyms

The following is a list of common terms and acronyms used within the Systems Engineering industry. While many of these terms are not mentioned within the body of this guide, they are nonetheless important to understanding this Systems Engineering Methodology (SEM). These terms are taken from several sources, including State of Michigan procedure documents, the State of Michigan Department of Information Technology Project Management Methodology, and the referenced sources contained in the SEM. If you need further information on any of the subjects in the following list, please consult the SEM, the variety of sources listed in the Websites listing (page viii) and the sources listed in the Resources and Reference lists contained throughout the SEM.

Acceptance criteria
The criteria that a software component, product, or system must satisfy in order to be accepted by the system owner or other authorized acceptance authority.

Acceptance process
The process used to verify that a new or modified system is fully operational and meets the system owner's requirements. Successful completion of the acceptance process results in the formal transfer of the system responsibilities from development to maintenance personnel.

Acceptance testing
Formal testing conducted to determine whether or not a software product or system satisfies its acceptance criteria and to enable the system owner to determine whether or not to accept the product or system.

Activity
A major unit of work to be completed in achieving the objectives of a project. An activity incorporates a set of tasks to be completed, consumes resources, and results in work products. An activity may contain other activities in a hierarchical manner. All project activities should be described in the Project Plan.

Algorithm
A finite set of well-defined rules for the solution to a problem in a finite number of steps. Any sequence of operations for performing a specific task.

Allocated requirements
The subset of the system requirements that are to be implemented within the scope of a given project, and forming the components of the system.

Anomaly
Anything observed in the operation or documentation of software and systems that deviates from expectations based on previously verified system or software products, or documents.

Application
Software or systems products designed to fulfill specific needs.

Assumption
A condition that is taken to be true without proof or demonstration.

Audit
An independent examination of a work product to assess compliance with specifications, standards, quality or security requirements, contractual agreements, or other predetermined criteria.

Baseline
A set of configuration items (hardware, software, documents) that has been formally reviewed and agreed upon, that serves as the basis for further development, and that can be changed only through formal change control procedures.

Baselined requirements
The set of project requirements that have been approved and signed off by the system owner during the Requirements Definition Stage. The system design is based on these requirements. The baselined requirements are placed under configuration control.

Code
Computer instructions and data definitions expressed in a development language or in a form that is output by an assembler, compiler, or other translator.

Code generator
A software tool that accepts as input the requirements or design for a computer program and produces source code that implements the requirements or design.

Code review
A meeting at which software code is presented to project personnel, managers, users, or other functional areas for review, comment, or approval.

Component
One of the parts that make up a system. A component may be hardware, software, or firmware and may be subdivided into other components.

Computer-Aided Software Engineering (CASE)
The use of computers to aid in the systems engineering process. May include the application of software tools for software design, requirements tracing, code production, testing, document generation, and other systems engineering activities.

Configuration control
An element of configuration management consisting of the evaluation, coordination, approval/disapproval, and implementation of changes to configuration items after formal establishment of their configuration identification.

Configuration Control Board (CCB)
A group of people responsible for evaluating and approving/disapproving proposed changes to configuration items, and for ensuring implementation of approved changes.

Configuration item
An aggregate of hardware, software, or documentation components that are designated for configuration management and treated as a single entity in the configuration management process.

Configuration Management
(1) A discipline that effectively controls and manages all modifications to system components, product, or system. Technical and administrative processes and tools are used to identify and document the functional and physical characteristics of the configuration items, manage and track changes to those items, record and report change processing and implementation status, and verify compliance with specified requirements.
(2) A Software Engineering Institute Capability Maturity Model Integrated key process area designed to establish and maintain the integrity of the software work products throughout the project's lifecycle.

Constraint
A restriction, limit, or regulation that limits a given course of action or inaction.

Construction Stage
The period of time in the project/product lifecycle during which a product is created from the design specifications. Testing is performed on the individual software units/components that have been coded, or on the combination of coded and purchased components, (e.g., as a COTS package.)

Cost estimate
A formal estimate of the cost to develop and support a project. Estimates should reflect all activities such as design, development, coding, testing, distribution, service, support of the product, staffing, training and travel expenses, subcontractor activities, contingencies,; and cost for external services (e.g., technical documentation production and Quality Assurance audits and reviews).

Deliverable
A work product that is identified in the Project Plan and is formally delivered to the system owner and other project stakeholders for review and approval.

Terms used in Describing Deliverables:

Initial – the Stage indicated is the first time the deliverable is worked on
Final – all work on the deliverable has been completed during the Stage indicated
Revised – the deliverable was worked on during the Stage indicated

Dependency
A relationship of one task to another where the start or end date of the second task is related to the start or end date of the first task.

Design
The process of defining the architecture, components, interfaces, and other characteristics of a system, product, or component.

Design specification
A document that describes the design of a software component, product, or system. Typical contents include architecture, control logic, data structures, input/output formats, interface descriptions, and algorithms.

Developer's Reference Manual
A work product deliverable that provides information necessary to maintain or modify components for a given computer system. Typically described are the equipment configuration, operational characteristics, coding features, input/output features, and compilation or assembly features of the computer system.

Feasibility
The degree to which the requirements, design, or plans for a software product or system can be implemented under existing constraints.

Functional area
Any formally organized group involved in the development and maintenance of systems or the support of development and maintenance efforts, or other group whose input is required to successfully implement a systems project. Examples of functional areas include systems engineering services, technical writing, quality assurance, security, and telecommunications.

Functional Design Stage
The period of time in the project lifecycle during which the designs for architecture, software components, interfaces, and data are created, documented, and verified to satisfy project requirements.

Functional requirement
A requirement that specifies a function that a software component, product, or system must be able to perform.

Functional Test Plan
A plan for testing each function across one or more units. The plan describes how the functional testing occurs and the test procedure/test cases that will be used. The plan includes procedures for creating the test environment that allows all functions to be executed, the entry and exit criteria for starting and ending the function test period, and the schedule followed for starting and ending each test.

Functional test procedures
Procedures for each function or combination of functions to be tested. Procedures fully describe how the function is tested. Expected output from each test procedure is identified to compare the planned output to actual output.

Functional testing
Testing conducted to evaluate the compliance of a software product with specified functional requirements. Testing that focuses on the outputs generated in response to selected inputs and execution conditions.

Glueware
Integration software that provides the proper interface for the component (i.e., wrappers) being integrated and serves as a mediator for its interactions with other components.

Hardware
Physical computer and other equipment used to process, store, or transmit computer programs or data.

Hierarchy
A structure in which components are ranked into levels of subordination.

Implementation requirements
A requirement that supports the development and maintenance concepts and approaches in the areas of operating environment, conversion, installation, training, and documentation.

Incremental development
A development technique in which requirements definition, design, implementation, and testing occur in an overlapping, iterative (rather than sequential) manner, resulting in incremental completion of the overall system or product.

Information Engineering
A development methodology where models are created to improve the users' ability to understand and define the functions and flow of information within their organization. A business model is developed to identify the key areas of interest for the business, the tasks required for each area, and the activities that make up each task. The business model prioritizes and identifies top management goals and then establishes the information needs necessary to reach those goals. A data model is developed to describe the data and the relationships among data. The data model further divides the business model into user-defined relationships (e.g., entity relationship model).

Initiation and Planning Stage
The initial stage in the project lifecycle during which the system owner/users' needs and expectations are identified, the feasibility of the project is determined, and the Project Charter and Project Plan are developed.

Inspection
A static analysis technique that relies on visual examination of development products to detect errors, violations of development standards, and other problems. Code inspection and design inspection are two types.

Integration testing
An orderly progression of testing in which software components are combined and tested to evaluate the interaction between them.

Integrity
The degree to which a software component, product, or system prevents unauthorized access to, or modification of, computer programs or data.

Interactive analysis and design
A development methodology that uses facilitated team techniques, such as Joint Application Development or Rapid Application Development, to rapidly develop project requirements that reflect the users' needs in terminology that the users understand. Group facilitation techniques are especially important when several user organizations have unique project requirements that are specific to their mission and goals.

Interface requirement
A requirement that specifies an external item with which a software product or system must interact, or that sets forth constraints on formats, timing, or other factors caused by such an interaction.

Interface testing
Testing conducted to evaluate whether system components pass data and control correctly to one another.

Interview technique
A technique for the identification, analysis, and documentation of the project requirements. The project team conducts a series of interviews with users to identify the users' perceived IT functional needs, analyzes the information gathered during the interviews, and develops the requirements.

Key Process Area
Software engineering processes identified by the Software Engineering Institute Capability Maturity Model Integrated where a project team should focus its efforts to achieve consistently high quality software products.

Lifecycle
See Project Lifecycle.

Maintenance
The process of supporting a software product or system after delivery to maintain operational status, correct faults, improve performance or other attributes, or adapt to a changed environment.

Menu-driven
Pertaining to a system or mode of operation in which the users direct the software through menu selections.

Methodology
A collection of methods, procedures, and standards that defines an integrated synthesis of engineering approaches to the development of a work product.

Milestone
A scheduled event for which an individual or team is accountable and that is used to measure progress.

Module
A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading. A logically separable part of a program.

Module testing
Testing of individual software modules or groups of related modules to verify the implementation of the design.

Organization
An organization is a unit within a company or other entity within which projects are managed as a whole. Examples of an organization include, MDIT, a contractor organization, a program (e.g., RW, or a laboratory.) All projects within an organization share a common top-level manager and common policies.

Performance requirement
A requirement that imposes conditions on a functional requirement (e.g., a requirement that specifies the speed, accuracy, or memory usage with which a given function must be performed).

Platform
A specific computer and operating system on which a software product or system is developed or operated.

Portability
The ease with which a software component, product, or system can be transferred from one hardware or software environment to another.

Procedure
A written description of a course of action to be taken to perform a given task.

Process
An ordered set of steps performed for a given purpose. Processes define or control the development of the project work products. The use of processes will ensure a consistent methodology across all platforms in producing the lifecycle deliverables.

Product
See Work product.

Project
An undertaking requiring concerted effort that is focused on developing or maintaining a specific software product or system. A project has a distinct beginning and end, and has its own funding, cost accounting, and delivery schedule.

Project file
A central repository of material and artifacts pertinent to a project. Contents typically include all work products, memos, plans, technical reports, and related items.

Project lifecycle
Covers all activities conducted within the scope of an entire project, from project startup to project closeout.

Project Management Methodology
The State of Michigan document that provides governance for all State of Michigan projects.

Project Management Plan
See Project Plan.
Synonymous with software development plan and project plan.

Project Manager
The individual with total responsibility for all activities of a project. The project manager plans, directs, controls, administers, and regulates a project.

Project Plan
A document that describes the technical and management approach to be followed for a project. The plan typically describes the work to be done, the resources required, the methods to be used, the procedures to be followed, the schedules to be met, and the way the project will be organized.

Project planning
A Software Engineering Institute Capability Maturity Model Integrated key process area designed to establish reasonable plans for performing systems engineering and for managing the project.

Project team
The project manager, analysts, developers, and other staff assigned as the core group for a project. The project team may include representatives of the other functional areas (e.g., technical writer and telecommunications expert) responsible for contributing to the development, installation, and maintenance of the software product.

Project Test Plan
Defines the what, how, where, and when about all test activities required to assure that the software product or system will perform satisfactorily for all users. As a minimum, the plan should include descriptions for unit testing, integration testing, system testing, and acceptance testing.

Project tracking and oversight
A Software Engineering Institute Capability Maturity Model Integrated key process area designed to provide adequate visibility into actual project progress so that management can take effective actions when the project's performance deviates significantly from the plans.

Prototyping
A technique for developing and testing a preliminary version of the software product (either as a whole or in modular units) in order to emulate functionality without such encumbering features as error handling, help messages, security controls, and other utilities that are not part of the design logic. This allows the project team to test the overall logic and workability of required functions and provides a model by which the project team and users can jointly determine if the software requirements meet the intended objectives. Prototyping is often used in conjunction with interactive analysis and design techniques.

Pseudocode
A combination of development language constructs and natural language used to express a computer program design.

Quality assurance
A process designed to provide management with appropriate visibility into the systems engineering processes being used by the project team and the work products being built. One of the Software Engineering Institute Capability Maturity Model Integrated Level 2 key process areas.

Rapid Prototyping
A type of prototyping in which emphasis is placed on developing prototypes earlier in the development process to permit early feedback and analysis in support of the development process.

Reference
A document(s) or other material that is useful in understanding more about an activity.

Regression testing
Selective retesting of a software or system component to verify that modifications have not caused unintended effects and that the software or system component still complies with its specified requirements.

Reliability
The ability of a software or system component to perform its required functions under stated conditions for a specified period of time.

Requirement
A condition or capability needed by a system owner/user to solve a problem or achieve an objective. A condition or capability that must be met or possessed by the software product or system to satisfy a contract, standard, specification, or other formally imposed documents.

Requirements analysis
The process of analyzing and understanding the scope and feasibility of identified requirements; of developing a preliminary plan to arrive at a detailed definition of system, hardware, or software requirements; and of crystallizing a preliminary system solution.

Requirements Definition Stage
The period of time in the project lifecycle during which the requirements for an IT product are defined and documented.

Requirements management
A process designed to establish a common understanding between the system owner/users and the project team regarding the system owner/users' software and system requirements. This understanding forms the basis for estimating, planning, performing, and tracking the project's activities throughout the lifecycle. One of the Software Engineering Institute Capability Maturity Model Integrated Level 2 key process area

Requirements Specification
A work product deliverable that specifies the manual and automated requirements for a software product or system in non-technical language that the system owner/users can understand. Typically included are functional requirements, performance requirements, and interface requirements. Describes in detail what will be delivered in the product or system release.

Retirement
Permanent removal of a system or software product from its operational environment.

Reusability
The degree to which a software module or other work product or system component can be used in more than one computer program or software system.

Reverse Engineering
A development methodology in which the software development process is performed in reverse. The technique involves the examination of an existing software product that has characteristics that are similar to the desired product. Using the existing code as a guide, the requirements for the product are defined, analyzed, and abstracted all the way back to specifications. Any required code changes can be made based on a specification-like format. Ideally, the specifications would be edited and passed to a code generator that would trigger automatic documentation and revisions. Once testing is complete, the revised code is placed into production.

Risk management
An approach to problem analysis that is used to identify, analyze, prioritize, and control risks.

Software
Computer programs, procedures, and associated documentation and data pertaining to the operation of a software product or system.

Software Quality Assurance
See Quality Assurance.

Specification
A document that specifies in a complete, precise, verifiable manner the requirements, design, behavior, and other characteristics of a software component, product, or system.

Spiral development model
A software development process in which the constituent activities, typically requirements analysis, design, coding, integration, and testing are performed iteratively until the software product is complete.

Stage
A partition of the project lifecycle that reduces a project to manageable pieces and represents a meaningful and measurable set of related tasks that are performed to obtain specific work products.

Stakeholder
The State of Michigan individual(s) with decision-making authority over a project or group of projects.

Standard
Mandatory requirements employed and enforced to prescribe a disciplined, uniform approach to software and systems development and maintenance.

Structured analysis
An analysis technique that uses a graphical language to build models of software products or systems. The four basic features in structured analysis are data flow diagrams, data dictionaries, procedure logic representations, and data store structuring techniques.

System
(1)A collection of hardware, software, firmware, and documentation components organized to accomplish a specific function or set of functions.
(2) A product and the documentation, hardware, and communications needed to implement and operate the product and accomplish a specific function or set of functions.

System Design Document
A work product deliverable that describes the solution to the automation task as described by the requirements. Contains sufficient detail to provide necessary direction for writing the Program Specifications and allows developers maximum technical freedom.

System Design Stage
A stage in the lifecycle model during which the designs for the software product or system architecture, software components, interfaces, and data are refined and expanded to the extent that the design is sufficiently complete to be implemented.

Systems engineering
An inter-disciplinary approach and means to enable the realization of successful systems.

Systems Engineering Methodology
The MDIT methodology that identifies the processes, activities, tasks, management responsibilities, and work products that are required for each system development and maintenance project. Deviations from the methodology require the approval of all stakeholders who have approval rights on the project. A key objective of the methodology is to provide measurable, repeatable processes to assure that project development and maintenance methodologies are consistent throughout the MDIT information technology environment.

System owner
The enterprise unit that funds and has approval authority for the project. Typically, system owners are also system users.

System testing
Testing conducted on a complete, integrated software product or system to evaluate compliance with its specified requirements.

Task
The smallest unit of work subject to management accountability. A task is a well-defined work assignment for one or more project team members. Related tasks are usually grouped to form activities.

A task is the lowest level of work division typically included in the Project Plan and Work Breakdown Structure.

Telecommunications
The science and technology of communications by electronic transmission of impulses, as by telephone or e-mail.

Test bed
An environment containing the hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test.

Test case
A set of test inputs, execution conditions, and expected results that are developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.

Test criteria
The criteria that a software product or system component must meet in order to pass a given test.

Test design
Documentation specifying the details of the test approach for a software or system feature or combination of features and identifying the associated tests.

Test documentation
Documentation describing plans for, or results of, the testing of a system component or product. Documents typically include test case specifications, test incident reports, test logs, test plans, test procedures, and test reports.

Test item
A system component that is the object of testing.

Test log
A chronological record of all relevant details about the execution and results of a test.

Test plan
A document specifying the scope, approach, resources, and schedule of intended testing activities. The plan identifies test items, the features to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning.

Test procedure
Detailed instructions for the setup, execution, and evaluation of the results for a given test case.

Test report
A document that describes the conduct and results of the testing carried out for a software or system component or product.

Testing
An activity in which a software or system component or product is executed under specified conditions, the results are observed and recorded, and an evaluation is made.

Testing Stage
The period of time in the project lifecycle in which the components of a system are evaluated and integrated, and the product is evaluated to determine whether or not the requirements have been satisfied.

Traceability
The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor relationship to one another.

Transaction analysis
A technique used to derive structured charts for a software product that will process transactions. Transaction analysis is used to divide complex data flow diagrams into smaller, simpler data flow diagrams--one for each transaction that the product or system will process. Structure charts are developed from the simple data flow diagrams. The individual structure charts for the separate transactions are then combined to form one large structure chart that is very flexible and can accommodate user changes.

Unit
A separately testable element specified in the design of a computer system or software component. A software or system component that is not subdivided into other components.

Unit testing
Testing of individual hardware or software units or groups of related units. The isolated testing of each flowpath of code with each unit. The expected output from the execution of the flowpath should be identified to allow comparisons of the planned output against the actual output.

Usability
The ease with which a user can learn to operate, prepare inputs for, and interpret outputs of an IT product or system.

User
Within the context of information systems, the general population of individuals who use a software product or system. User activities can include data entry; read only; add, change and delete capabilities; querying; and report generation.

User interface
An interface that enables information to be passed between a user and hardware or software components of a computer system.

User manual
A document that presents the information necessary to use a software product or system to obtain desired results. Typically described are product or component capabilities, limitations, options, permitted inputs, expected outputs, possible error messages, and special instructions.

Validation
The process of evaluating software or systems at the end of the development process to assure compliance with established software and system requirements.

Verification
The process of evaluating a software product or system to determine whether or not the work products of a stage of the project lifecycle fulfill the requirements established during the previous stage.

Walkthrough
An analysis technique in which a team of subject matter experts review a segment of code, documentation, or other work product, ask questions, and make comments about possible errors, violation of development standards, and other problems.

Work product
Any tangible item that results from a project function, activity, or task. Examples of work products include process descriptions, plans, procedures, computer programs, and associated documentation, which may or may not be intended for delivery to the system owner and other project stakeholders.

Bibliography: The following materials were used in the preparation of the glossary.

1. Carnegie Mellon University, Software Engineering Institute, *Capability Maturity Model: Guidelines for Improving the Software Process,* Addison Wesley Longman, Inc., 1994.

2. Software Engineering Institute, Software Process Maturity Questionnaire, Capability Maturity Model, version 1.1, Carnegie Mellon University, Pittsburgh, 1994.

3. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard Glossary of Software Engineering Terminology,* IEEE Std 610.12-1990, The Institute of Electrical and Electronics Engineers, Inc., New York, 1990.

4. Webster's II New Riverside University Dictionary, Houghton Mifflin Company, Boston, 1984.

Appendix B – SEM Templates, Checklists and Guides

In Order by Chapter / Phase

The following is a list of the SEM templates/checklists and SEM process guides referenced in the SEM. They are ordered as they appear in the chapters of this document.

Chapter 1: None.

Chapter 2: None.

Chapter 3: Maintenance Plan (SEM-0301)
Software Configuration Management Plan (SEM-0302)

Chapter 4: Requirements Traceability Matrix (SEM-0401)
Requirements Specification (SEM-0402)
Requirements Management Checklist (SEM-0403)

Chapter 5: Functional Design Document (SEM-0501)

Chapter 6: Conversion Plan (SEM-0601)
Test Plan (SEM-0602)
Test Report (SEM-0603)
System Design Document (SEM-0604)
System Design Checklist (SEM-0605)
Software Testing Checklist (SEM-0606)

Chapter 7: Transition Plan (SEM-0701)
Installation Plan (SEM-0702)
Training Plan (SEM-0703)
Training Checklist (SEM-0704)

Chapter 8: Integration and System Testing Checklist (SEM-0801)
Error Reporting and Tracking Checklist (SEM-0802)
Pre-Acceptance Checklist (SEM-0803)
Testing Package Checklist (SEM-0804)
User Acceptance Checklist (SEM-0805)

Chapter 9: None.

The Structured Walkthrough Process Guide and the Stage Exit Process Guide are utilized in all SEM stages.

Page inserted for consistency in section start points.

Appendix C – Investigate Alternatives

Activity: **C-1. Investigate Alternatives**

Responsibility: Project Manager/Team

Description: Software and hardware alternatives are reviewed during the overall project Initiation and Planning Stage and are used to formulate preliminary platform options. Use the project objectives, scope, and high-level requirements as the basis for conducting research and investigating resources.

Sample Questions: The following is a list of sample questions that can be used to help investigate software and hardware alternatives and provide recommendations that fit within the budget and goals for the project.

- Is the alternative feasible within time and resource constraints and limitations?
- Is there at least one technically feasible IT solution for the project?
  - If a project is well defined and has no automation issues, a single straightforward IT solution may sufficiently demonstrate cost and technical feasibility.
  - Where IT issues have been identified, technical alternatives should be associated with each proposed solution.

Work Products: Refer to each task for applicable work products.

Review Process: Refer to each task for applicable review processes.

Tasks: The following tasks are involved in investigating alternatives and will assist in completing the Enterprise Architecture (EA) Solution Assessment document. Once the EA Core Team develops their own instruction set for the Solution Document, these sections will be removed from the SEM.

C.1.1 Investigate Software Alternatives
C.1.2 Investigate Hardware Alternatives
C.1.3 Formulate Platform Options

Touch Points: The following touch points are involved in investigating alternatives:

- Enterprise Architecture (EA) - Solution Assessment document
- Contracts and Procurement – procurement related documents, including DIT-0153 Bid Information Sheet, DIT-0015b, Statement of Work
- Office of Enterprise Security (OES) – Security liaison sign-off on Security Plan initiation

Task: **C.1.1 Investigate Software Alternatives**

Description: When the software to be used for the project has not been predetermined by the system owner's existing IT environment, software available within the State of Michigan and the commercial marketplace should be investigated. In the Initiation and Planning Stage, the investigation of software alternatives is geared to determining project feasibility.

Unless the cost effectiveness of developing custom-built software to meet mission needs is clear and documented, all sources of reusable code, applications, and commercial-off-the-shelf (COTS) software must be investigated on a site and Department-wide basis prior to making a decision to custom build code for the project. This practice ensures the most cost-effective and efficient use of resources, and will decrease the number of duplicative and overlapping software systems. The choice to develop a customized application should be balanced against the availability of other solutions; and the project cost, resources, and time constraints.

Software Alternatives: Information on software products or modules can be obtained from the MDIT Bureau of Strategic Policy, other Government agencies, and private industry via the Internet. The following is a list of software alternatives that should be considered.

- Adapt existing software in use within the Department.
- Adapt existing software in use within other Government agencies, such as local units of government that use and interface with SOM systems.
- Adapt mainframe or minicomputer source code obtained from existing State of Michigan systems.
- Purchase commercial-off-the-shelf (COTS) software.
- Reuse existing modules of code.
- Adapt reusable code to fit the new application.
- Develop a custom-built software product.

Note: Medium and small information systems engineering efforts are often restricted to the system owner's existing software. This should not preclude the potential cost savings of re-engineering existing software modules rather than custom building the entire software system.

Task: **C.1.2 Investigate Hardware Alternatives**

Description: When the hardware to be used for the project has not been predetermined by the system owner's existing IT environment, investigate hardware available within the State of Michigan and through the commercial marketplace. In the Initiation and Planning Stage, the investigation of hardware is geared to determining project feasibility.

Factors to Consider: The following is a list of factors that should be considered when identifying hardware alternatives.

- Availability and cost of hardware
  - Shareable hardware
  - Underutilized hardware
  - New procurement
- Architecture compliance
- Current and future communications needs
- Computer security requirements of the system
- Volume of data
- Importance of data to the MDIT-Department-Agency missions
- Importance of data to the user organization's mission and to job performance
- Potential growth to serve more users
- Potential growth to serve more locations
- Interface to other systems or organizations
- Conformance to SOM standards such as networking and open systems

Note: Medium and small systems engineering efforts are often restricted to the system owner's or user sites' existing hardware.

Task: **C.1.3 Formulate Platform Options**

Description: Use the information collected about software and hardware alternatives to formulate preliminary platform options. The purpose of identifying platform options early in the project lifecycle is to assure that at least one technically feasible and cost-effective approach exists to satisfy the project objectives. If more than one platform option is feasible, identify the benefits, costs, assumptions, constraints, dependencies, and risks associated with each option. Platform options must be compatible with the State of Michigan target Enterprise Architecture (EA).

No platform decisions are made at this time. Detailed technical solutions are premature prior to defining the product requirements. The platform alternatives information gathered in the Initiation and Planning Stage is revisited in the Functional Design Stage. At this time a final recommendation is developed by the project team, including the appropriate Infrastructure Services staff, and presented to the system owner. The system owner is responsible for making the final platform decision.

Work Product: Develop an EA Solutions Assessment for use in the Infrastructure Services Request process and the Hosting Solution Document. Place a copy of these documents in the Project File.

Review Process: Conduct a structured walkthrough to ensure that the most viable platform options have been identified.

Appendix D – List of Acronyms

CASE	Computer-Aided Software Engineering
CMMI	Capability Maturity Model Integrated
COTS	Customized Off-The-Shelf
EA	Enterprise Architecture
FCA	Functional Configuration Audit
IT	Information Technology
LAN	Local area network
MDIT	Michigan Department of Information Technology
OES	Office of Enterprise Security
PCA	Physical Configuration Audit
PMM	Project Management Methodology
SCM	Software Configuration Management
SEI	Software Engineering Institute (at Carnegie-Mellon)
SEM	Systems Engineering Methodology
SOM	State of Michigan
SUITE	State Unified Information Technology Environment
SWT	Structured Walkthrough